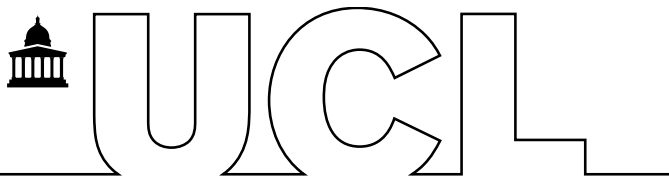
LONDON'S GLOBAL UNIVERSITY



Global Optimisation for Dynamic Systems using Novel Overestimation Reduction Techniques

by

Carlos Perez Galvan

A Dissertation Submitted to the
Department of Chemical Engineering of
University College London
in Partial Fulfilment of the Requirements for the Degree of

Doctor of Philosophy in Chemical Engineering

Centre for Process Systems Engineering
Department of Chemical Engineering
University College London

June 2017

'I, Carlos Perez Galvan confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.'

_____

Carlos Perez Galvan

To Liliana, Jorge, Hugo,

my mother Rosa and my father Rodolfo

Abstract

The optimisation of dynamic systems is of high relevance in chemical engineering as many practical systems can be described by ordinary differential equations (ODEs) or differential algebraic equations (DAEs). The current techniques for solving these problems rigorously to global optimality rely mainly on sequential approaches in which a branch and bound framework is used for solving the global optimisation part of the problem and a verified simulator (in which rounding errors are accounted for in the computations) is used for solving the dynamic constraints. The verified simulation part is the main bottleneck since tight bounds are difficult to obtain for high dimensional dynamic systems. Additionally, uncertainty in the form of, for example, intervals is introduced in the parameters of the dynamic constraints which are also the decision variables of the optimisation problem. Nevertheless, in the verified simulation the accumulation of trajectories that do not belong to the exact solution (overestimation) makes the state bounds overconservative and in the worst case they blow up and tend towards $\pm\infty$.

In this thesis, methods for verified simulation in global optimisation for dynamic systems were investigated. A novel algorithm that uses an interval Taylor series (ITS) method with enhanced overestimation reduction capabilities was developed. These enhancements for the reduction of the overestimation rely on interval contractors (Krawczyk, Newton, Forward-Backward) and model reformulation based on pattern substitution and input scaling. The method with interval contractors was also extended to Taylor Models (TM) for comparison purposes. The two algorithms were tested on several case studies to demonstrate the effectiveness of the methods. The case studies have a different number of state variables and system parameters and they use uncertain amounts in some of the system parameters and initial conditions. Both of the methods were also used in a sequential approach to address the global optimisation for dynamic systems problem subject to uncertainty.

The simulation results demonstrated that the ITS method with overestimation reduction techniques provided tighter state bounds with less computational expense than the traditional method. In the case of the forward-backward contractor additional constraints can be introduced that can potentially contribute significantly to the reduction of the overestimation. Similarly, the novel TM method with enhanced overestimation reduction capabilities provided tighter bounds than the TM method alone. On the other hand, the optimisation results showed that the global optimisation algorithm with the novel ITS method with overestimation reduction techniques converged faster to a rigorous solution due to the improved state bounds.

Acknowledgements

First, I would like express my thanks to my supervisor Professor Ian David Lockhart Bogle for helping me complete this project, for the many useful and critical discussions, the words of encouragement and for his commitment from the beginning until the end of this doctorate.

I am extremely grateful to my family for always supporting me at every stage in my life in every sense. Their encouragement and support gave me the force I needed to have a positive attitude every day. Their almost daily calls made be concious that they are always there for me, even for the silliest of the things. To my siblings Liliana, Jorge and Hugo and my parents Rosa Galván and Rodolfo Pérez I dedicate this thesis. I am also very grateful to my cousins Ana Cecilia González and Yesenia Aguirre for their love and support.

I would like to acknowledge my sponsors. Without the financial of the Mexican Council of Science and Technology (CONACyT), University College London (UCL) through the Faculty of Engineering Sciences, the Mexican Secretary of External Affairs (SRE) and "Centro Kappa de Conocimiento, S.C" this project would not have been possible.

My thanks also go to my friends form Mexico, Roberto Benavente and Dr Lourdes Morales (Lulu) and to my friends from the Product and Process Systems Engineering office of the Department of Chemical Engineering at UCL. I thank Charalampos Christodoulou (Harry), Dr Cristian Triana, Andres Calderon, Mariya Koleva (Masha), Matthew Darby (Matt), William Ashworth (Will), Folashade Akinmolayan (Shade), Justin Siefker, David Lorenzo, Miguel Vieira and Dr Quentin Meyer for their friendship and academic support.

Table of contents

List of Figures

List of Tables

List of Mathematical Notation

The next list describes the mathematical notation that will be later used within the body of the document

$[y]$	Scalar interval
$\underline{y}$	Lower endpoint of scalar interval
$\overline{y}$	Upper endpoint of scalar interval
$\hat{y}$	Midpoint of an scalar interval
$w([y])$	Width of an scalar interval
h_j	Size of the time step j
$[\tilde{\mathbf{y}}_j^0]$	Range of the solution for a given set of ODEs at time step j
$[\tilde{\mathbf{y}}_j]$	*a priori* enclosure at time step j
$[\mathbf{y}_j]$	Tight enclosure at time step j
$[\mathbf{y}_{j+1}]$	Tight enclosure at time step $j+1$
$f^{[i]}$	i^{th} Taylor coefficient of a function
$\mathcal{T}_f$	Taylor model of a function
$\mathcal{P}_f$	Polynomial part of the Taylor model of a function
$[R_f]$	Interval remainder term part of the Taylor model of a function
$\hat{\mathcal{P}}_f$	Centred polynomial part of the Taylor model of a function
$\hat{\mathcal{T}}_f$	Centred Taylor model of a function
$[R_f^c]$	Centred interval remainder term part of the Taylor model of a function
$[\mathcal{T}_f]$	Bounds on the Taylor model of a function
$[\mathcal{P}_f]$	Bounds on polynomial part of the Taylor model of a function
$\mathbf{I}$	Identity matrix
$diag(\mathbf{A})$	Matrix with the elements replaced by 0 except in the diagonal
$extdiag(\mathbf{A})$	Matrix with only the diagonal elements replaced by 0

Chapter 1

Introduction

Natural phenomena can be described in a mathematical way by means of models. Dynamic models aim to describe the vast amount of these phenomena that change over time. In science and engineering dynamic models are used to simulate, predict or understand systems of interest through the use of ordinary differential equations (ODEs) and differential-algebraic equations (DAEs). One is usually concerned with the best operating conditions or the best trajectory that the system can reach; as this knowledge allows us to save resources, reduce time and cost, and improve quality and/or safety.

To accomplish this, optimisation problems constrained by ODEs or DAEs are formulated with the objective of finding the best trajectory that satisfies the constraints by modifying some controls. Such problems are called dynamic optimisation problems. Chemical engineering applications where optimal operating conditions are sought are parameter estimation, model predictive control, operating profiles of batch processes, safety and environmental impact analysis, etc.

A difficulty arising in dynamic optimisation is the presence of nonconvexity due to the nonlinear combination of terms participating in the dynamic system. These nonconvexities lead to multiple local minima which make finding the global optimum a daunting task. However, the global optimum is usually desired since it represents the benefits as the ones aforementioned: saving resources, reducing time and cost, and improving quality and/or

safety.

Furthermore, addressing this problem in a rigorous way such that the solution set is guaranteed to be true in a computational environment is of special importance in safety critical applications where having bounds on the dynamic variables of interest represent the line between having a dangerous or acceptable concentration in biological systems for example [2] or these bounds can also draw the line between wanted or unwanted temperature or pressure that could compromise the safety of the operation or yield undesired by-products [76]. Therefore, solving rigorous global optimisation for dynamic systems problems is of importance in industry and engineering.

Two classes of discretisation approaches are usually employed for solving the dynamic optimisation problem. In the first class, the control and state variables are discretised (this class is also known as full discretisation) leaving an optimisation problem that can be solved using a nonlinear programming (NLP) technique. In the second class, only the control variables are discretised (partial discretisation) leading to a problem that has two main parts: an integration of the dynamic constraints and an optimisation of a bound-constrained problem.

Solving a dynamic optimisation problem in a rigorous way means that the algorithmic implementation is able to account for the rounding and truncation error of floating point numbers in a systematic way. In a simultaneous approach, one must make sure that every single computation in the NLP solver is rigorous. Similarly, in a sequential approach rigorous computations are needed when solving the dynamic system and when solving the bound constrained optimisation problem. This thesis considers an approach of the latter form since solving the former means that rigorous bounds are available for the relaxed, fully discretised problem, which might not hold true for the dynamic problem. However, more work in rigorous simultaneous dynamic optimisation is needed to show its advantages.

In this context, the objective of this thesis is to develop efficient methods for computing guaranteed bounds on all the possible solutions resulting from ranges in the inputs of a

given ODE model. Using these methods, this thesis also aims to rigorously solve optimisation problems with embedded ODEs to global optimality.

The first part of the objectives is related with obtaining guaranteed bounds for the set that can be reached at time t by a solution of a given ODE system resulting from a consistent choice of initial conditions and system parameters, i.e., the reachable set. In this case, the choice of the initial conditions and system parameters usually is a range of values such as an interval. The type of solvers able to address these problems in a rigorous way are commonly called verified or validated.

The second part is concerned with using the developed methods in an optimisation solver that exhaustively searches for the global optima in the whole decision space in a systematic way using a branch and bound framework.

The next sections expand on the two main objectives of the thesis. Section 1.1 reviews work on computing guaranteed bounds on the reachable set of a given ODE system, provides a motivation for pursuing research in this area and presents the contributions of the thesis. Similarly, Section 1.2 reviews work on global optimisation for dynamic systems and provides a motivation and contributions of the thesis.

1.1 Computing Guaranteed Bounds for ODEs

Obtaining bounds on the dynamic variables is a key step in the solution of dynamic optimisation problems using a sequential approach. Integrating the dynamic system with inputs in the form of ranges allows us to construct upper and lower bounds on the state trajectories. This information can in turn be used in an optimisation procedure that discards regions of the decision space that are infeasible. Furthermore, the implementation of such algorithms in safety critical applications can be enabled by freeing the computations from round-off and truncation errors (granting rigorousness to the calculations) and thus only true solutions are given by the computer results.

In the past, different approaches to solve this problem have been studied. Because of the need to be rigorous in the computations, techniques that involve set computations such as interval arithmetic [43], and more recently ellipsoidal arithmetic [81], have been adopted. In addition, due to the potential application in global optimisation algorithms, methods to construct convex relaxations, such as McCormick relaxations [41, 71], have been investigated.

An interval Taylor series (ITS) method for computing the reachable set with ranges in the initial conditions of a given ODE system was proposed by Moore [42]. In his method he described a process that uses a Taylor series to verify the solution. He also noticed the presence of overestimation due to the wrapping effect (described in Chapter 2) in an oscillating problem. Since then, several modifications have been proposed to the method in order to reduce the wrapping effect. In particular, Kruckeberg [30], Eijgenraam [16] and Lohner [36] used modifications to avoid direct evaluations of one of the products known to be the main contributor of overestimation in Taylor series. From these methods, the QR decomposition [36] turned out to be one of the best alternatives to tackle the wrapping effect. Nedialkov et al. [47] provides a review of these methods and discusses their characteristics. The VNODE-LP software package developed by Nedialkov [46] is yet another variation of the ITS that uses an Hermite-Obreschkoff approach to address the overestimation. McCormick relaxations (convex and concave relaxations) have been used in the remainder term of the interval Taylor series (ITS) method in an approach called discretise then relax [66].

Approximate enclosure with validated enclosures have been developed as well. Rauh et al. [61] described a verified ODE solver that uses an approximate enclosure, Picard iterations to verify the enclosure and consistency tests to improve it. The software package ValEncIA-IVP is based on this method. Rauh et al. [63] expanded the previous method by using an interval Newton technique to address DAEs in ValEncIA-IVP. Moreover, Rauh and Auer [60] extended the previous method by implementing an exponential enclosure for asymptotically stable systems.

Differential inequalities methods for enclosing systems of ODEs with intervals in the initial conditions and system parameters have been developed. Papamichail and Adjiman [52] described a method for bounding quasi-monotone ODEs with ranges in the system parameters using differential inequalities to construct upper and lower ODE systems enclosing the original system. Singer and Barton [74] presented a method using differential inequalities and physical information and applied it to ODEs that are not necessarily quasi-monotone. Using the previous method, they also proposed a procedure to construct affine convex and concave relaxations of the ODEs using McCormick relaxations. Scott et al. [72] developed a method for constructing convex and concave relaxations for nonlinear ODEs using the generalised McCormirck relaxation technique [71]. Scott and Barton [70] proposed a method that uses a combination of the affine relaxations of Singer and Barton [74] and the nonlinear theory of Scott et al. [72] to derive tighter bounds for nonlinear ODEs.

Methods based on Taylor models have also been published. Berz and Makino [7] described a method for computing the bounds of ODE systems, using remainder differential algebra (RDA) and the Schauder theorem to express the dependence on initial values and time. Lin and Stadtherr [34] described a method using a two stage Taylor series procedure similar to the interval Taylor series (ITS) approach [42, 47]. In their method they computed Taylor models of Taylor coefficients using the RDA for the propagation of the operations and for the wrapping effect control they used the QR decomposition and the parallelepiped procedures. VSPODE, a state of the art software package, incorporates these algorithms.

Sahlodin and Chachuat [65] developed a method similar to Lin and Stadtherr [34] called McCormick-Taylor model. The main difference was in propagating Taylor models with convex/concave remainder using the generalised McCormick relaxation technique [71] instead of interval remainders. The convergence properties of this method were analysed in Bompadre et al. [9].

Houska et al. [24] devised a method for verified ODE solution using Taylor models with ellipsoidal remainder bounds. For asymptotically stable systems, their algorithm is guaran-

teed to be stable on infinite time horizons within some uncertainty.

The methods of differential inequalities and Taylor models have also been used in collaboration. Chachuat and Villanueva [13] described a method for propagating Taylor models and McCormick-Taylor models using the theory of differential inequalities. Villanueva et al. [80] provided a mechanism for constructing the support function of a convex enclosure of the reachable set, which allows the use of the theory of differential inequalities for propagating convex (intervals, ellipsoids) and nonconvex sets (Taylor models with interval or ellipsoidal remainder bounds).

1.1.1 Motivation

Since early works, the ability to represent the solution set, which might be nonconvex, with *simpler* convex sets has been a challenge. The reason for this is that these convex sets overestimate the true solution set and accumulate points that do not belong to the true solution set at every time step. The overestimation produced can lead to overconservative state bounds that make it difficult to distinguish the true solution. In the worst case the overestimation make the state bounds tend towards $\pm\infty$ and the integration stops.

Furthermore, because of the need for having computations that account for ranges in the inputs of a given ODE, the approaches based on set computations such as interval and ellipsoidal arithmetic do not directly translate to real arithmetic operations. For example, subtracting to identical intervals does not produce an interval of zero width but an interval with twice the original width. Interval arithmetic operations usually overestimate the true range of an expression due to the dependency problem (Section 2.3.1).

In interval analysis, particularly in the area of steady-state nonlinear optimisation and constraint satisfaction problems, there have been successful works that do reduce the overestimation of interval evaluated functions (Section 2.1.2). These techniques are based on finding a valid inclusion of the overestiamated solution by applying fixed-point methods, constraint propagation, set inversion, etc (Section 2.4).

As already mentioned, there have been many developments in the area of verified integration of ODEs with ranges in the inputs. However, no particular research has been done on the reduction/elimination of overestimation at each time step. Most of the aforementioned methods do not use techniques such as interval fixed-point methods or interval constraint propagation to reduce the overestimation of the bounds. Since the main purpose of such techniques is the reduction of the overestimation, this thesis considers these techniques in Chapters 3, 4, 5 and 6.

1.1.2 Contributions

The first part of the thesis aims to reduce the overestimation in verified integration algorithms making use of techniques from interval nonlinear optimisation and constraint propagation. In Chapter 3 two fixed-point interval contractors (interval Krawczyk and Newton contractor) are used in an interval Taylor series method. Similarly, Chapter 4 makes use of a constraint propagation contractor and a collaboration of the constraint propagation contractor and the fixed-point contractor in an interval Taylor series method. Chapter 5 extends the use of the fixed-point contractors to Taylor models with interval remainder. Part of the material in these chapters has appeared in the following peer-reviewed publications:

[58] C. Perez-Galvan and I. D. L. Bogle. Global optimisation for dynamic systems using interval analysis. *Computers & Chemical Engineering*, 2017. doi: http://dx.doi.org/10.1016/j.compchemeng.2017.02.028. URL `http://www.sciencedirect.com/science/article/pii/S0098135417300923`

[2] W. Ashworth, C. Perez-Galvan, N. Davies, and I. D. L. Bogle. Liver function as an engineering system. *American Institute of Chemical Engineers Journal*, 62(9):3285–3297, 2016. doi: 10.1002/aic.15292. URL `http://onlinelibrary.wiley.com/doi/10.1002/aic.15292/full`

[57] C. Perez-Galvan and I. D. L. Bogle. Reduction of the overestimation in verified simulation by model reformulation. In Z. Kravanja and M. Bogataj, editors, *26th European Symposium on Computer Aided Process Engineering*, pages 1857–1862, Slovenia, 2016. Elsevier

[54] C. Perez-Galvan and I. D. L. Bogle. Comparison between Interval Methods to Solve Initial Value Problems in Chemical Process Design. In J. J. Klemeš, P. S. Varbanov, and P. Y. Liew, editors, *24th European Symposium on Computer Aided Process Engineering*, volume 33 of *Computer Aided Chemical Engineering*, pages 1405–1410, Budapest, 2014. Elsevier. ISBN 9780444634344. doi: 10.1016/B978-0-444-63455-9.50069-6. URL `http://www.sciencedirect.com/science/article/pii/B9780444634559500696`

1.2 Rigorous Global Optimisation for Dynamic Systems

Several chemical engineering researchers have devoted their efforts to solve the guaranteed global optimisation problem for dynamic systems. They have used complete search methods such as branch and bound frameworks to make sure no solution is left out and they have also focused on finding bounds for the dynamic variables. Therefore, their algorithms rigorously find all global optima within bounds or are at least they are able to provide a theoretical guarantee that the global optima have been found.

A branch and bound framework was used with the αBB method [1] in a sequential approach and applied to four different optimal control problems including the optimal control of batch and semi-batch processes [19], and to parameter estimation problems to determine reaction kinetic constants from time series data [18]. In principle the method of [18, 19] provides a guaranteed global optimum. However, the rigorous underestimators needed are hard to obtain and here only sampling approaches were proposed. Another sequential approach implementing a branch and bound framework was developed with a convex underestimating procedure [52] which they applied to parameter estimation, chemical kinetics and modelling and optimal control problems. Some years later, again using a sequential approach and a branch and bound framework, Papamichail and Adjiman [53] used McCormick relaxations

and constant and affine bounds in the solution of parameter estimation and optimal control problems. The approach used by Papamichail and Adjiman [52, 53] is computationally expensive in constructing tight affine underestimators and overestimators. Singer and Barton [73] presented a sequential approach using another branch and bound framework for problems with an integral objective function. This algorithm implements differential inequalities and McCormick relaxations to construct the convex/concave relaxations and the method was applied to parameter estimation and optimal control problems. Rauh et al. [62] presented a global optimisation method for discrete-time and continuous-time systems applied to a mechanical positioning system. In the continuous-time system their algorithm uses a prototype implementation of an interval extension of Taylor series as a verified solver. The guaranteed solution has also been considered for mixed-integer dynamic optimisation. Chachuat et al. [14] presents a review on these methods focusing on systems with embedded ODEs and branch and bound frameworks. They stress out the importance of tight state relaxations and the use of heuristics in the global optimisation algorithm. The dynamic optimisation problem has also been addressed using Taylor models in a sequential approach and a branch and bound framework with focus on the tightness of the ODEs state bounds. This method uses Taylor models method with an interval remainder term and was applied to several parameter estimation problems [32]. Later, they applied the same method but this time with a branch and reduce approach using a domain reduction technique [33] and the applications were an optimal control and a final time formulation of the oil shale pyrolysis problems. The latter method was extended to account for inequality path constraints in a rigorous way [83] and was applied to three semi-batch reactor problems.

1.2.1 Motivation

In safety critical applications, there is the need of making sure that the process is operating safe at all times. For example, in the regulation of insulin in glucose homeostasis there are upper and lower limits that if surpassed can cause conditions of hyperglycaemia and hypoglycaemia. Also in the regulation of temperature of a system with a limit safety temperature, when the temperature surpasses the limit the reaction in the system can produce undesired by-products or compromise the stability of the reaction system.

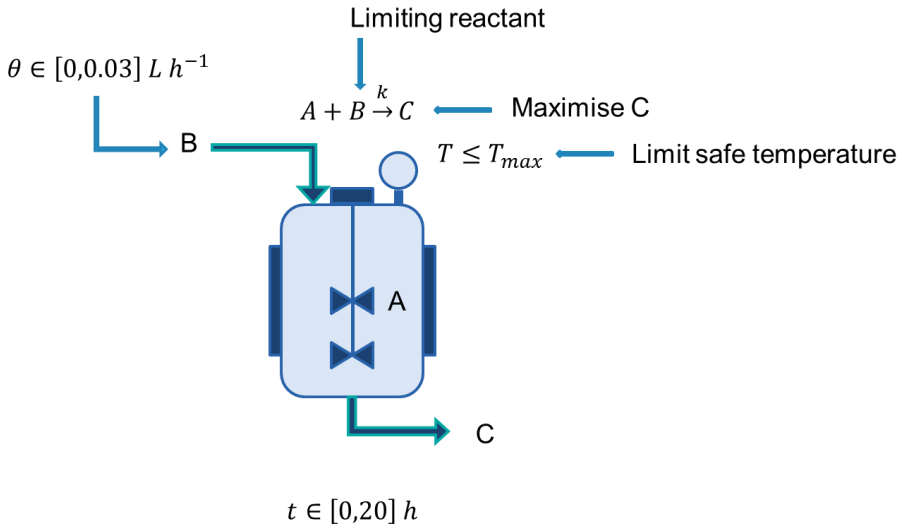


Figure 1.1: Exothermic reactor with safety constraint

Illustrative example. Consider the batch reactor in Figure 1.1. The process in the figure aims to maximise the concentration of C at final time. In this reaction, the reactant B is the limiting reactant and is being input to the reactor as a flow rate θ for which a range is known. The problem is also constrained by a safety constraint in the form of a maximum temperature and must be obeyed at all times otherwise the operation can become dangerous since the reaction is highly exothermic. Figure 1.2 shows all the possible reachable trajectories by the variable temperature and the safety constraint. Since the reaction is favoured by the temperature, the optimal point is the one with highest temperature at final time. Local solvers can obtain a local minimum for the problem as shown by the dashed line in the figure. However, this thesis aims to obtain the global optimum of the problem, which is represented by the continuous line. This thesis also aims to provide upper and lower bounds on the global optima to make sure the process operates within the safe limits at all times.

Rigorous bounds are needed to address these kinds of problems, that means the implementation must be guaranteed to yield true results (guaranteed upper and lower bounds on the global optima).

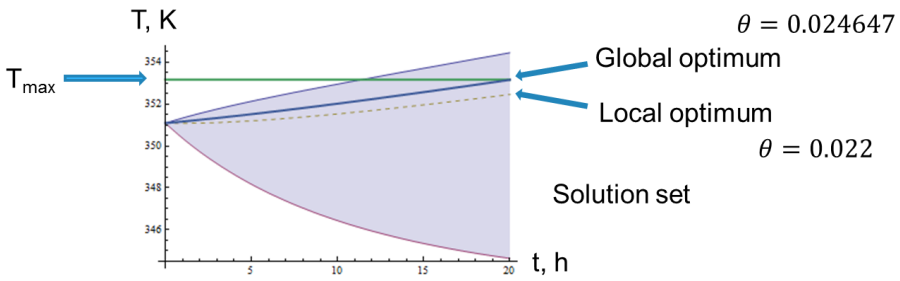


Figure 1.2: Possible trajectories of the variable temperature for the input range of reactant B

As previously discussed, there are sequential approaches that use different techniques to address the dynamic optimisation problem that are based on some verified integration with control or reduction of the overestimation. The main motivation of this part is to find out how a verified solver with overestimation reduction techniques (based on fixed-point interval contractors) affects the performance of a global optimisation for dynamic systems algorithm.

1.2.2 Contributions

The verified solver developed in Chapter 3 is used in Chapter 6 in a global optimisation algorithm. Case studies from optimal control and guaranteed parameter estimation demonstrate the effectiveness of the method. The material in these chapters has appeared in the following peer-reviewed publications:

[58] C. Perez-Galvan and I. D. L. Bogle. Global optimisation for dynamic systems using interval analysis. *Computers & Chemical Engineering*, 2017. doi: http://dx.doi.org/10.1016/j.compchemeng.2017.02.028. URL `http://www.sciencedirect.com/science/article/pii/S0098135417300923`

[56] C. Perez-Galvan and I. D. L. Bogle. Dynamic Global Optimisation Methods for Determining Guaranteed Solutions in Chemical Engineering. In P. M. Pardalos, A. Zhigl-

javsky, and J. Zilinskas, editors, *Advances in Stochastic and Deterministic Global Optimization*. Springer International Publishing, 1 edition, 2016. ISBN 978-3-319-29973-0. doi: 10.1007/978-3-319-29975-4. URL http://www.springer.com/gp/book/9783319299730

[55] C. Perez-Galvan and I. D. L. Bogle. Deterministic Global Dynamic Optimisation using Interval Analysis. In K. V. Gernaey, J. K. Huusom, and R. Gani, editors, *12th International Symposium on Process Systems Engineering and 25th European Symposium on Computer Aided Process Engineering*, volume 37 of *Computer Aided Chemical Engineering*, pages 791–796. Elsevier, 2015. ISBN 9780444634290. doi: 10.1016/B978-0-444-63578-5.50127-4. URL http://www.sciencedirect.com/science/article/pii/B9780444635785501274

1.3 Thesis Organisation

The thesis is organised in chapters, each starting with a small abstract, followed by an introduction that reviews the relevant literature, a problem statement, a development of the methods, numerical case studies and concluding remarks. At the end of the thesis, conclusions for the thesis as a whole are provided as well. The sequence of the chapters is as follows: Chapter 1 presents an introduction with contributions of the thesis. Chapter 2 describes the most important topics to be used in the thesis. Chapter 3 develops a verified integration method consisting of an interval Taylor series (ITS) with interval fixed-point contractors (interval Krawczyk and Newton/Gauss-Seidel contractors) for the reduction of the overestimation. In Chapter 4 a method for verified integration that implements a constraint propagation contractor in an ITS method for the reduction of the overestimation, was developed. In Chapter 5 the same approach as Chapter 3 is used but with as Taylor models method instead of the ITS method. Chapter 6 uses the verified solver from Chapter 3 in a global optimisation algorithm that implements a branch and bound framework to solve dynamic optimisation problems. Chapter 7 concludes the thesis and presents remarks for future work. Finally, appendices for relevant topics are provided.

Chapter 2

Background

In this chapter key concepts that are used throughout the thesis are described. The first section provides some notions on interval analysis as they are used in Chapters 3, 4, 5 and 6. Concepts on a two-stage process for verified integration are also provided as Chapters 3 and 5 use these techniques. In Chapters 3 and 5 the main objective is to tackle overestimation using interval contractors. Therefore, overestimation and contractors are explained in general terms. Finally, notions on global optimisation using a branch and bound algorithm are outlined as these are useful in Chapter 6.

2.1 Interval Analysis

Interval analysis uses closed intervals as the main number to do computations with. A closed interval is defined as

$$[a] = [\underline{a}, \overline{a}] = \{x \in \mathbb{R} \mid \underline{a} \leq x \leq \overline{a}\} \tag{2.1}$$

Numbers like π can be represented by the interval $[3.1415, 3.1416]$ and therefore avoiding the need to approximately represent the number by rounding it. Furthermore, when the interval upper and lower endpoints are equivalent, the interval is called degenerate ($[a] = [a, a]$).

The basic arithmetic operations have been defined for intervals:

Addition

$$[a] + [b] = [\underline{a} + \underline{b}, \overline{a} + \overline{b}] \tag{2.2}$$

Subtraction

$$[a] - [b] = [\underline{a} - \overline{b}, \overline{a} - \underline{b}] \tag{2.3}$$

Multiplication

$$[a] \times [b] = [\min\{\underline{ab}, \underline{a}\overline{b}, \overline{a}\underline{b}, \overline{a}\overline{b}\}, \max\{\underline{ab}, \underline{a}\overline{b}, \overline{a}\underline{b}, \overline{a}\overline{b}\}] \tag{2.4}$$

Division

$$[a]/[b] = [a] \times 1/[b] = [a] \times [1/\overline{b}, 1/\underline{b}] \tag{2.5}$$

where $0 \notin [b]$

Some useful interval functions are:

Midpoint

$$\hat{a} = \frac{\underline{a} + \overline{a}}{2} \tag{2.6}$$

Width

$$w([a]) = \overline{a} - \underline{a} \tag{2.7}$$

Absolute value

$$| [a] | = \max\{| \underline{a} |, | \overline{a} |\} \tag{2.8}$$

Also set operations such as the union and the intersection are defined for intervals in the following way

Union

$$[a] \cup [b] = [\min\{\underline{a}, \underline{b}\}, \max\{\overline{a}, \overline{b}\}] \tag{2.9}$$

Intersection

$$[a] \cap [b] = [\max\{\underline{a}, \underline{b}\}, \min\{\overline{a}, \overline{b}\}] \tag{2.10}$$

2.1.1 Interval Vectors

An interval vector or box is an ordered n-tuple of intervals

$$[\mathbf{a}] = ([a_1], [a_2], ..., [a_n]).$$

The usual midpoint, width and absolute value functions apply to interval vectors in a component-wise fashion. Similarly, the union and intersection are also defined component by component. In this thesis, the bold type notation is used to denote vector- and matrix-valued quantities.

2.1.2 Interval Functions

Consider a vector of real-valued functions $\mathbf{f}$ dependent on a vector variable $\mathbf{x}$ for which one is interested in knowing the range taken by $\mathbf{f}(\mathbf{x})$ as $\mathbf{x}$ varies in an interval $[\mathbf{x}]$. The goal is finding the image of the set $[\mathbf{x}]$ under the mapping $\mathbf{f}$

$$\mathbf{f}([\mathbf{x}]) = \{\mathbf{f}(\mathbf{x}) \mid \mathbf{x} \in [\mathbf{x}]\} \tag{2.11}$$

An enclosure of this set can be obtained interval extensions of functions.

Interval Extension

An interval extension ($[\mathbf{f}]$) refers to a function whose domain has been expanded to include intervals ($[\mathbf{x}] = [\underline{\mathbf{x}}, \overline{\mathbf{x}}]$) as well as degenerate intervals ($[\mathbf{x}] = [\mathbf{x}, \mathbf{x}]$)

$$\mathbf{f}(\mathbf{x}) = [\mathbf{f}]([\mathbf{x}, \mathbf{x}]).$$

For example, the interval extension of

$$\mathbf{f}(\mathbf{x}) = \mathbf{x}(1 - \mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^n$$

is

$$[\mathbf{f}]([\mathbf{x}]) = [\mathbf{x}](1 - [\mathbf{x}]), \quad [\mathbf{x}] = [\underline{\mathbf{x}}, \overline{\mathbf{x}}].$$

One is often interested in the smallest box (interval vector) that contains the set image of a particular function. An interval extension is minimal if for any $[\mathbf{x}]$, $[\mathbf{f}]([\mathbf{x}])$ is the smallest box that contains $\mathbf{f}([\mathbf{x}])$, this extension will be denoted by $[\mathbf{f}]^*([\mathbf{x}])$. Usually, the interval extension of a function is not equivalent to $[\mathbf{f}]^*([\mathbf{x}])$, that is

$$[\mathbf{f}]([\mathbf{x}]) \supseteq [\mathbf{f}]^*([\mathbf{x}]) \supseteq \{\mathbf{f}(\mathbf{x}) \mid \mathbf{x} \in [\mathbf{x}]\}.$$

This is because in interval arithmetic, there is a lack of distributivity and additive and multiplicative inverses. Thus, interval extensions are not unique as they heavily rely on the expression $\mathbf{f}$. When one simply evaluates a function using interval arithmetic operations, the extension is called natural interval extension. Very useful interval arithmetic libraries have been developed for interval analysis.For example, INTLAB has been developed for Matlab by Rump [64], Mathematica has its own built in library as well. In C++, we have PROFIL/BIAS by [29] and Ibex by [12].

2.2 Verified Integration Based on Taylor Forms

The general building blocks of two-stage algorithms for verified integration are described in this section. In particular, the interval Taylor series and the Taylor models methods make use of automatic differentiation to compute the Taylor coefficients. In general terms, they are comprised by two stages similar to a predictor-corrector method. The first stage is used to perform the validation and uniqueness of the solution in which a preliminary and often conservative enclosure is obtained. The second stage is used for tightening the solution. These three aspects are discussed in the next sections.

2.2.1 Automatic Generation of Taylor Coefficients

When one tries to compute enclosures of ODE systems using an interval Taylor series or Taylor models method, Taylor coefficients are needed in the intermediate steps. One way to compute them is by using automatic differentiation. Consider the ODE system

$$\dot{y}(t) = f(y), \;\; y(t_j) = y_j \tag{2.12}$$

which is assumed to be i times continuously differentiable. Its first and second Taylor coefficients can be obtained by

$$\begin{aligned} f^{[0]}(y_j) &= y_j \\ f^{[1]}(y_j) &= f(y_j) \end{aligned} \tag{2.13}$$

and the rest can be obtained by the sequence

$$f^{[i]}(y_j) = \frac{1}{i}\left(\frac{\partial f^{[i-1]}}{\partial y} f\right)(y_j), \quad for \;\; i \geq 1 \tag{2.14}$$

As an example, some Taylor coefficients for common algebraic expressions are provided. Here, u and v are analytic functions and i is the order of the Taylor coefficient

$$\begin{aligned} (u \pm v)^{[i]} &= (u)^{[i]} \pm (v)^{[i]} \\ (u \times v)^{[i]} &= \sum_{j=0}^{i} (u)^{[j]}(v)^{[i-j]} \\ (u^a)^{[i]} &= \left(\frac{1}{u}\right) \sum_{j=0}^{i-1} \left(a - \frac{j(a+1)}{i}\right)(u)^{[i-j]}(u^a)^{[j]} \\ \left(\frac{u}{v}\right)^{[i]} &= \left(\frac{1}{(v)^{[0]}}\right)\left\{(u)^{[i]} - \sum_{j=1}^{i}(v)^{[j]}\left(\frac{u}{v}\right)^{[i-j]}\right\} \\ (e^u)^{[i]} &= \sum_{j=0}^{i-1}\left(1 - \frac{j}{i}\right)(e^u)^{[j]}(u)^{[i-j]} \\ (\ln u)^{[i]} &= \left(\frac{1}{u}\right)\left\{(u)^{[i]} - \sum_{j=1}^{i-1}\left(1 - \frac{j}{i}\right)(u)^{[j]}(\ln u)^{[i-j]}\right\} \end{aligned} \tag{2.15}$$

In this thesis, the Taylor coefficients have been obtained using the C++ library FADBAD++ [4].

2.2.2 *A priori* Enclosure

Verified solvers for ODEs based on the two-stage process such as the interval Taylor series method and the Taylor models method need of a stage for the validation and uniqueness of the solution. This section presents two approaches for computing a preliminary enclosure that provides conservative bounds that are guaranteed to include the true solution of an ODE system, also known as, the *a priori* enclosure. The first of the approaches (the constant

enclosure) is used to illustrate how the preliminary state bounds are obtained and later on it is expanded to a better version which is the high order enclosure.

Constant Enclosure

The problem considered in this section is to find a step size $h_j > 0$ and an *a priori* enclosure $[\tilde{y}(t)]$ such that $y_j \in [y_j]$.

$$\dot{y}(t) = f(t, \theta), \;\; y(t_j) = y_j \tag{2.16}$$

has a unique solution $y(t) \in [\tilde{y}(t)]$ for $[t_j, t_{j+1}]$

The Picard-Lindelhöf operator is used to prove that there exists a unique fixed point.

$$(Ty)(t) = y_j + \int_{t_0}^{t} f(y(s))ds \tag{2.17}$$

Let $\mathcal{C}^0(\mathcal{X}, \mathcal{Y})$ be the set of continuous functions from $\mathcal{X} \subseteq \mathbb{R}$ to $\mathcal{Y} \subseteq \mathbb{R}^n$ and consider the set of continuous functions on $[t_j, t_{j+1}]$ with ranges on $[\tilde{y}_j^0]$, $\mathcal{Z} = \{u \in \mathcal{C}^0([t_j, t_{j+1}], [\tilde{y}_j^0])\}$. Applying the Picard-Lindelhöf operator to a function $y^0 \in \mathcal{Z}$ we obtain

$$\begin{aligned} (Ty^0)(t) &= y_j + \int_{t_j}^{t} f(y^0(s), \theta)ds \\ (Ty^0)(t) &\in y_j + \int_{t_j}^{t} f([\tilde{y}_j^0], \theta)ds \\ (Ty^0)(t) &\subseteq y_j + [0, h_j]f([\tilde{y}_j^0], \theta) \end{aligned} \tag{2.18}$$

According to Nedialkov et al. [47], the operator T has a unique fixed point in $\mathcal{Z}$ if $T\mathcal{Z} \subseteq \mathcal{Z}$ which is a solution of (2.16) for $[t_j, t_{j+1}]$. If

$$[\tilde{y}_j] = [y_j] + [0, h_j]f([\tilde{y}_j^0], \theta) \subseteq [\tilde{y}_j^0] \tag{2.19}$$

holds, then $T\mathcal{Z} \subseteq \mathcal{Z}$ and (2.16) has a unique solution $y(t; t_j, y_j, \theta)$ that satisfies

$$y(t; t_j, y_j, \theta) \in [y_j] + [0, h_j]f([\tilde{y}_j^0], \theta) = [\tilde{y}_j] \tag{2.20}$$

for all $t \in [t_j, t_{j+1}]$ and all $y_j \in [y_j]$. Now, since $f([\tilde{y}_j]) \subseteq f([\tilde{y}_j^0])$, the solution also satisfies

$$y(t; t_j, y_j, \theta) \in [y_j] + [0, h_j] f([\tilde{y}_j], \theta) \tag{2.21}$$

for all $t \in [t_j, t_{j+1}]$ and all $y_j \in [y_j]$. By starting with an initial wide interval $[\tilde{y}_j^0]$, that includes the solution $[y_j]$, the *a priori* enclosure $[\tilde{y}_j]$ can be obtained provided a small enough step size h_j is used in (2.19). The step size should as small as to make sure the inclusion $[\tilde{y}_j] \subseteq [\tilde{y}_j^0]$ holds. If it does not, the step size is reduced until the previous inclusion holds. The only problem of (2.20) is that h_j is restricted to small step sizes as in Euler method where the step size is restricted to the equation: $y_{j+1} = y_j + h_j f(y_j)$. However, it is desired to have time steps sizes as large as possible to reduce the computational time. The next section describes an approach that computes a larger step size making use of a high order method.

High Order Enclosure

Corliss and Rihm [15] proposed a theorem for finding a high order enclosure and enable the use of bigger steps. Their method applies a Taylor expansion to (**??**) and after integration it yields:

$$y(t; t_j, y_j, \theta) \in [\tilde{y}_j] = \sum_{i=0}^{k-1} (t - t_j)^i f^{[i]}(y_j, \theta) + (t - t_j)^k f^{[k]}([\tilde{y}_j^0], \theta) \subseteq [\tilde{y}_j^0] \tag{2.22}$$

for all $t \in [t_j, t_{j+1}]$ and all $\theta \in [\theta]$. Nedialkov et al. [48] described a method generalising the high order enclosure. If the inclusion

$$[\tilde{y}_j] = \sum_{i=0}^{k-1} [0, h_j]^i f^{[i]}([y_j], [\theta]) + [0, h_j]^k f^{[k]}([\tilde{y}_j^0], [\theta]) \subseteq [\tilde{y}_j^0] \tag{2.23}$$

holds, then $y(t; t_j, y_j, \theta)$ has a unique solution that satisfies

$$y(t; t_j, y_j, \theta) \in [\tilde{y}_j] \tag{2.24}$$

for all $t \in [t_j, t_{j+1}]$ and all $\theta \in [\theta]$. The main advantage of the high order enclosure method is that the step size can be bigger than in the constant enclosure method.

2.2.3 Tighter Enclosure

This algorithm computes a tighter enclosure for which $y(t_{j+1}; t_j, [y_j], \theta) \subseteq [y_{j+1}] \subseteq [\tilde{y}_j]$. The Taylor expansion of the solutions of (2.16) for times between t_j and t_{j+1} is

$$y_{j+1} = \sum_{i=0}^{k-1} h_j^i f^{[i]}(y_j, \theta) + h_j^k f^{[k]}(y(t_j + \tau), \theta), \quad for\ some\ \tau \in [0, h_j] \tag{2.25}$$

Moore [43] used the following equation to compute bounds for $\dot{y}(t) = f(y(t))\ y(t_0) = t_0 \in [y_0]$:

$$[y_{j+1}] = \sum_{i=0}^{k-1} h_j^i f^{[i]}([y_j]) + h_j^k f^{[k]}([\tilde{y}_j]) \tag{2.26}$$

However, the solution provided by this formula usually grows as the independent variable increases. One way to avoid this is by using the mean-value theorem on y

$$\begin{aligned} y_{j+1} = &\sum_{i=0}^{k-1} h_j^i \left(f^{[i]}(\hat{y}_j, \theta) + \frac{\partial f^{[i]}}{\partial y}(\eta(y(t_j) - \hat{y}_j) + \hat{y}_j, \theta)\{y(t_j) - \hat{y}_j\} \right) \\ &+ h_j^k f^{[k]}(y(t_j + \tau), \theta) \end{aligned} \tag{2.27}$$

for some $\eta \in [0, 1]$ where $\hat{y}_j \in [y_j]$.

After applying an interval extension to the previous equation, we can obtain an expression for computing the enclosure $[y_{j+1}]$ using only interval arithmetic (Chapter 3)

$$\begin{aligned} &= \sum_{i=0}^{k-1} h_j^i f^{[i]}(\hat{y}_j, \hat{\theta}) + \left\{ \sum_{i=0}^{k-1} h_j^i \frac{\partial f^{[i]}}{\partial y}([y_j], [\theta]) \right\} ([y_j] - \hat{y}_j) \\ &+ \left\{ \sum_{i=0}^{k-1} h_j^i \frac{\partial f^{[i]}}{\partial \theta}([y_j], [\theta]) \right\} ([\theta] - \hat{\theta}) + h_j^k f^{[k]}([\tilde{y}_j], [\theta]) \end{aligned} \tag{2.28}$$

where $\hat{y}_j = (\underline{y_j}, \overline{y_j})/2$, $\hat{\theta} = (\underline{\theta}, \overline{\theta})/2$ and $[\tilde{y}_j]$ is obtained in the first phase with (3.3).

On the other hand, if we obtain the Taylor models of the Taylor coefficients, parame-

ters and initial conditions, then an enclosure $[\mathcal{T}_{y_{j+1}}(\theta)]$ can be obtained using Taylor model arithmetic (Chapter 5)

$$\mathcal{T}_{y_{j+1}}(\theta) = \sum_{i=0}^{k-1} h_j^i \mathcal{T}_{f^{[i]}}(\hat{\mathcal{P}}_{y_j}(\theta), \theta) + \left\{ \sum_{i=0}^{k-1} h_j^i \mathcal{T}_{\frac{\partial f^{[i]}}{\partial y}}(\mathcal{T}_{y_j}(\theta), \theta) \right\} [\hat{R}_{y_j}] + h_j^k f^{[k]}([\tilde{y}_j], [\theta]) \quad (2.29)$$

where $\hat{\mathcal{P}}_{y_j}(\theta)$ is the centred polynomial part of $\mathcal{T}_{y_j}(\theta)$ and $[\tilde{y}_j]$ is obtained in the first phase with (3.3).

2.3 Overestimation in Verified Integration

The overestimation in verified integration is originated by several sources. The dependency problem, cancellation and the wrapping effect are form of overestimation. The dependency problem and cancellation are caused mainly because the interval arithmetic operations do not directly translate real arithmetic operations. On the other hand, the wrapping effect is due to the repeated enclosure of a solution set by a convex simpler set that accumulates overestimation at every time step.

2.3.1 Dependency Problem

The dependency problem arises in interval analysis because the method does not account for the dependency of multiple repetitions of the variables in a mathematical model. In other words, a variable repeated twice in a model is treated as two separate variables because in the definition of the interval multiplication rule, the left and right factors vary independently. The next example illustrates the dependency problem.

Example 1. Consider the following function: $f(x) = x^2 - x$ and the ranges of $[x] = [1, 2]$. The natural interval extension of f using $[x]$ yields

$$f([x]) = [x]^2 - [x] = [1, 4] - [1, 2] = [-1, 3]$$

However, if we rewrite the function as $f(x) = (x - \frac{1}{2})^2 - \frac{1}{4}$ and perform the natural interval extension again the result is different

$$f([x]) = ([x] - \tfrac{1}{2})^2 - \tfrac{1}{4} = [\tfrac{1}{4}, \tfrac{9}{4}] - \tfrac{1}{4} = [0, 2]$$

The first time the function was evaluated the true range of the function was overestimated but in the second case the exact range was obtained.

2.3.2 Cancellation

When the mathematical expressions contain at least one addition or subtraction, some overestimation is generated. Contrary to floating point arithmetic, the widths in interval arithmetic are additive instead of cancelling.

Example 2. Consider $[x] = [1, 2]$ with width $w([x]) = (\overline{x} - \underline{x}) = 1$. Performing the subtraction $[x] - [x]$ does not yield 0 but a result with twice as much the width of $[x]$. The same width is obtained if the intervals are added instead.

$$[x] - [x] = [-1, 1]$$

$$[x] + [x] = [2, 4]$$

2.3.3 Wrapping Effect

According to Lohner [35] it is the undesirable overestimation of a solution set of an iteration or recurrence which occurs if this solution set is replaced by a superset of some 'simpler' structure (e.g. an interval) and this superset is then used to compute enclosures for the next step which may eventually lead to an exponential growth of the overestimation. More generally, according to Neumaier [50], wrapping is the overestimation due to the depth of a computational graph describing the ODE system. This overestimation, is caused by long sequences of nested operations depending on a limited number of variables only, which also magnifies bounds on rounding errors and hence can give wide meaningless results even for problems with exact data.

Example 3. The next linear differential equations example taken from [47] illustrates the wrapping effect.

$$\begin{aligned} \dot{y}_1 &= y_2 \\ \dot{y}_2 &= -y_1 \end{aligned} \tag{2.30}$$

The solution to this problem with initial conditions at y_0 is given by

$$y(t) = \begin{pmatrix} \cos(t) & \sin(t) \\ -\sin(t) & \cos(t) \end{pmatrix} y_0 \tag{2.31}$$

In the first time step y_0 will be mapped by

$$\begin{pmatrix} \cos(t) & \sin(t) \\ -\sin(t) & \cos(t) \end{pmatrix}$$

into a rectangle of the same size. This rectangle (solution set) needs to be enclosed by an interval vector with sides parallel to the axes of the plane (y_1, y_2). As the integration proceeds, the solution set remains the same size and at every time step, the an interval vector encloses the solution set with sides parallel to the axes y_1 and y_2. Thus, at every time step the solution set is being rotated without having its size changed but the enclosing rectangles will become larger and larger as they wrap the solution set (Figure 2.1).

2.4 Constraint Satisfaction Problems

Consider a vector function $\mathbf{f}$ with domain in the interval vector $[\mathbf{x}]$ as $\mathbf{f}(\mathbf{x}) = 0$. Then a constraint satisfaction problem can be formulated as

$$(\mathbf{f}(\mathbf{x}) = \mathbf{0}, \mathbf{x} \in [\mathbf{x}]) \tag{2.32}$$

with solution $\mathcal{S} = \{\mathbf{x} \in [\mathbf{x}] \mid \mathbf{f}(\mathbf{x}) = \mathbf{0}\}$.

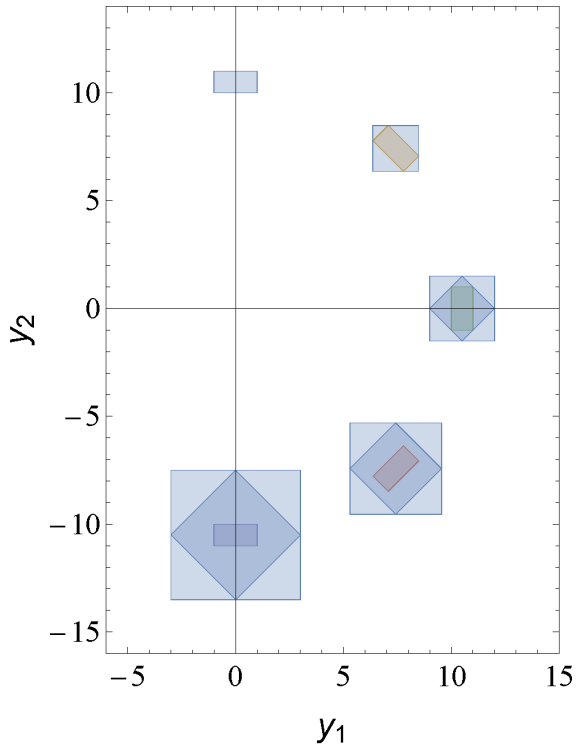


Figure 2.1: Wrapping effect illustration

Table 2.1: Contractors used in this thesis

Contractor	**Chapter**	**Reference**
Krawczyk	3	[49], [27]
Newton/Gauss-Seidel	3, 4, 5 & 6	[49], [27]
Forward-backward	4	[27], [5], [25]

Normally, when we obtain the natural interval extension of $\mathbf{f}$, the range of the function is overestimated. Contractors ($\mathcal{C}$) are operators that offer the possibility of replacing the domain $[\mathbf{x}]$ by a smaller domain $\mathcal{C}([\mathbf{x}])$ such that the true solution set remains unchanged. If $\mathcal{S} \subset \mathcal{C}([\mathbf{x}]) \subset [\mathbf{x}]$, then it is said that (2.32) has been contracted.

2.4.1 Interval Contractors

Table 2.1 provides the type of contractors that are considered in this thesis as well as the method in which they are based and the chapter in which they are discussed.

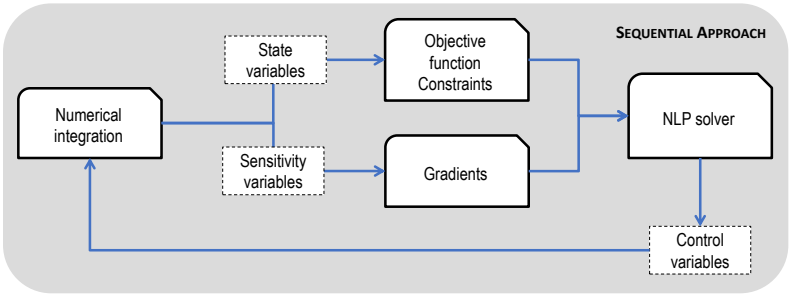
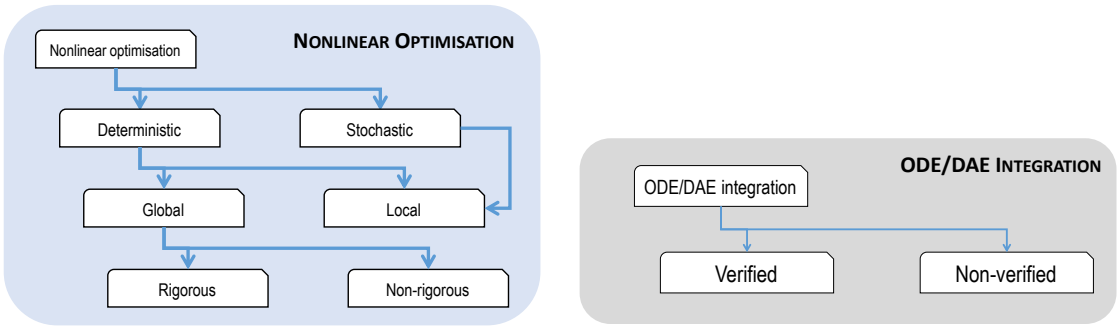


Figure 2.2: Illustration of a sequential approach for global optimisation of dynamic systems



(a) (b)

Figure 2.3: A classification for global optimisation and verified integration

2.5 Rigorous Global Optimisation for Dynamic Systems

In Figure 2.2 an illustration of the global optimisation for dynamic systems approach to follow is given. The figure illustrates a sequential approach for solving dynamic optimisation problems. In a sequential approach, two main algorithmic parts are used, which are the nonlinear optimisation algorithm (Figure 2.3a) and the ODE/DAE integrator (Figure 2.3b). In the case of the global optimisation algorithm (Section 2.5.2), this work focuses on the boxes in Figure 2.3a that include deterministic, global and rigorous methods that are guaranteed to include the global optimum within upper and lower bounds. On the other hand, in the ODE/DAE integration algorithm (Section 2.2), we stress out the use of the subclass verified methods from Figure 2.3b that are able to construct the trajectories of the dynamic system also within upper and lower bounds. In this thesis the methods for verified integration used are: interval Taylor series (ITS) and Taylor models. As it will be seen during the thesis, the main obstacle in the verified integration is the reduction of the overestimation.

2.5.1 Sequential Approach for Dynamic Optimisation

As already stated in the introduction (Chapter 1), the method for global optimisation in this thesis to be used is a sequential approach in which only the control variables are discretised. In the method, an integration of the dynamic system provides the information for evaluating the objective function and constraints. The optimum of the optimisation problem is then obtained by a rigorous branch and bound framework.

The advantage of these methods is that at every iteration, a feasible solution of the ODE system is obtained by integration for some control values, i.e., a feasible path method. Additionally, sequential methods generate smaller discrete problems than simultaneous methods [10].

In an optimisation problem, it is always desirable to have the gradients on the objective function and constraints with respect to the controls if the problem is differentiable. By having to integrate at each iteration in the optimisation problem, it is enough to obtain the Jacobian of the ODE system to use it as the Jacobian of the sensitivity equations. The solution of the sensitivity equations provides the gradient of the state variables with respect to the controls which is used to evaluate the gradients of the objective function and constraints.

2.5.2 Branch and Bound for Global Optimisation

A bound-constrained global optimisation problem needs to be solved after bounds for the state variables become available. Since we are interested in rigorous upper and lower bounds for the global optima the algorithm considered heavily relies on interval arithmetic. The following steps describe the global optimisation algorithm presented in Chapter 6.

Initialisation. The decision space is initialised with an interval vector big enough so as to ensure the global optima is inside the space and not in the edges or outside it. The upper bound is also initially set to $+\infty$.

Upper bound. The first approximation upper bound of the global optima can be a solution provided by a local or stochastic solver.

Bisection. Branch in the coordinate direction such that the width of the element to be branched is the maximum with respect to the elements in the current box.

Bounding. The verified ODE solver is called to obtain state bounds for each of the boxes generated that in turn serve to evaluate the objective function and constraints.

Upper bound update. The upper bound can be updated by comparing the upper bounds of the boxes just bisected and bounded and the solution obtained by the local or stochastic solver.

Working list. If the two boxes result of the bisection are not within the prescribed absolute or relative tolerance, then place these boxes in a list L in increasing order of minimum objective function value. Take the first box in this list (with the minimum value in the objective function) and go to the bisection step.

ε **convergence.** If the bisected box is within the prescribed absolute or relative tolerance, then place the boxes in order in a list C and if there are no more boxes to visit, then return with the lower bound of the first box in C and with list C. If there are more boxes to visit, then take the first box of list L, bound it and test if the lower bound is greater than the current upper bound, if so, return with the lower bound of the first box in C and with lists C and L.

Chapter 3

Interval Contractors in Interval Taylor Series

The present chapter deals with developing a method for verified ODE integration with overestimation reduction capabilities, suitable for a global optimisation algorithm. The chapter focuses on the interval Taylor series method with an embedded interval fixed-point contractors (Krawczyk and Newton/Gauss-Seidel contractors) that reduces the overestimation at each time step, i.e., it substitutes the state bounds for tighter state bounds that still include the true solution. The algorithm developed is tested in seven case studies from chemical and biological systems and is compared with a state-of-the-art solver.

3.1 Introduction

Dynamic simulation represents a key step in the deterministic solution for dynamic optimisation problems. When the objective is to rigorously obtain the best solution of the dynamic optimisation problem, a global optimisation technique is required as well as the ability to manage the round off and truncation errors in the integration stage, which can be done using a verified integration method for the solution of the dynamic system.

The solution of the initial value problems (IVPs) for ODEs plays a key role in the rigorous deterministic solution of dynamic optimisation problems since they provide the information

to evaluate the objective function and constraints. There are methods for this purpose which are mostly based on interval analysis or use some implementation of it. A number of researchers have developed several methods for the verified solution of IVPs for ODEs. One of the first attempts to find a verified solution for these systems was proposed by [42]. His idea consists of the use of the Picard-Lindelhof iteration in the system of ODEs and the Taylor expansion of this iteration. In his method, he provides a constant enclosure step to determine an *a priori* enclosure (coarse enclosure) of the solution.

Several other modifications to Moore's method have been made by other researchers including Eijgenraam [16]. The purpose of these modifications were to reduce the wrapping effect (Section 2.3.3) so Eijgenraam proposed several transformations. The modifications proposed by Lohner [36] also had an important impact as he devised a method involving a QR factorization of the matrix product responsible for the wrapping effect (Section 2.3.3). This method is still one of the best alternatives to tackle this problem. Nedialkov et al. [47] provide an excellent review about these methods.

Other alternatives to bound the ODE states with upper and lower bounds have been proposed, for example the Interval Hermite-Obreschkoff method by Nedialkov [46], the Taylor Models approach by Lin and Stadtherr [31] and by Makino and Berz [39], the differential inequalities approach by Scott and Barton [69] and the validated enclosure of an approximate solution by Rauh et al. [63]. McCormick relaxations (convex and concave relaxations) have been used in the remainder term of the interval Taylor series (ITS) method [66]. These relaxations have also been used in the Taylor Models method in the remainder term [65]. A kind of Taylor model with ellipsoidal remainder term has also been devised [24]. Fazal and Neumaier [20] computed state bounds using conditional differential inequalities. The method requires global optimisation subproblems that can make the method verified if a rigorous global optimisation solver is used.

However, there is still need for a method able to obtain tighter and more efficient bounds since solving solving dynamic optimisation problems of practical size remains an issue.

Considering all the previous methods, this work makes use of an ITS method because of its implementation simplicity. This method is prone to be modified and take new implementations in an easier manner especially in the development stage and when proposing new methods. The interval Taylor series has already been subject to some changes recently. It is the basis of the method of the popular software package VNODE [46] which also used the interval Hermite-Obreschkoff approach. McCormick relaxations have also been used in the method in Sahlodin and Chachuat [66]. In this chapter, we propose to implement interval contractors (Krawczyk and Newton/Gauss-Seidel) at each iteration of ITS method so as to reduce the overestimation generated. The next section describes the traditional ITS method.

3.2 Problem Formulation

We assume that the system can be described by an ODE model $\dot{\mathbf{y}}(t) = \mathbf{f}(t, \mathbf{y}(t), \boldsymbol{\theta})$, with $\mathbf{y}$, the vector of state variables and $\boldsymbol{\theta}$, the vector of system parameters. In this chapter the bold type notation is adopted to indicate vector-valued quantities and square brackets are used for interval-valued quantities unless otherwise specified. Just as in Chapter 2, the lower and upper endpoints of an interval $[x]$ are specified by $[\underline{x}, \overline{x}]$, the width or diameter of an interval is given by $w([x]) = \overline{x} - \underline{x}$ and the midpoint by $\hat{x} = (\overline{x} - \underline{x})/2$. The mathematical form of the problem that this method aims to solve is as follows:

$$\dot{\mathbf{y}}(t) = \mathbf{f}(t, \mathbf{y}(t), \boldsymbol{\theta}),\ \mathbf{y}(t_0, \boldsymbol{\theta}) = \mathbf{y}_0(\boldsymbol{\theta}),\ \mathbf{y}_0(\boldsymbol{\theta}) \in [\mathbf{y}_0],\ \boldsymbol{\theta} \in [\boldsymbol{\theta}] \tag{3.1}$$

where $t \in [t_0, t_f]$, $\boldsymbol{\theta}$ represent the time-invariant parameters with $[\boldsymbol{\theta}] = [\underline{\boldsymbol{\theta}}, \overline{\boldsymbol{\theta}}]$, $\mathbf{y}$ represent the vector of state variables, $\mathbf{y}_0$ are the initial conditions at time t_0 with $[\mathbf{y_0}] = [\underline{\mathbf{y}_0}, \overline{\mathbf{y}_0}]$. Here, $\mathbf{f}$ is assumed to be $(k-1)$ times continuously differentiable with respect to the state variables.

A solution of (3.1) from the initial condition $\mathbf{y}_j$ at t_j and parameter values $\boldsymbol{\theta}$ is denoted by $\mathbf{y}(t; t_j, \mathbf{y}_j, \boldsymbol{\theta})$. The solution set of (3.1) with initial conditions $[\mathbf{y}_j]$ at t_j and parameter values $[\boldsymbol{\theta}]$ is given by $\mathbf{y}(t; t_j, [\mathbf{y}_j], [\boldsymbol{\theta}]) = \{\mathbf{y}(t; t_j, \mathbf{y}_j, \boldsymbol{\theta}) \mid \mathbf{y}_j \in [\mathbf{y}_j], \boldsymbol{\theta} \in [\boldsymbol{\theta}]\}$. The objective of

this method is to compute an enclosure

$$[\mathbf{y}_j] \supseteq \mathbf{y}(t_j; t_0, [\mathbf{y}_0], [\boldsymbol{\theta}]) = \{\mathbf{y}(t_j; t_0, \mathbf{y}_0, \boldsymbol{\theta}) \mid \mathbf{y}_0 \in [\mathbf{y}_0], \boldsymbol{\theta} \in [\boldsymbol{\theta}]\} \tag{3.2}$$

at $t_j \in [t_0, t_f]$, $j = 1, \ldots N_s$. The bounding method used in this chapter consists of two stages. The first stage is the validation of existence and uniqueness of a solution (Section 3.3.1) in which also a suitable *a priori* enclosure and a time step are obtained. The second stage involves the computation of a tighter enclosure (Section 3.3.2) which consists on the use of a high order Taylor series to refine the solution obtained in the first stage.

3.3 Verified Integration of ODEs. Interval Taylor Series

In this section, an interval Taylor series (ITS) method has been used for obtaining bounds on the dynamic variables of the ODE models. The bounding method used in this chapter consists of two stages, the approach is similar to the one of the VNODE software package [46] except that the parametric dependency is being explicitly accounted for. The first stage is the validation of existence and uniqueness of a solution in which also a suitable *a priori* enclosure and a time step are obtained. The second stage involves the computation of a tighter enclosure which consists on the use of a high order Taylor series to refine the solution obtained in the first stage.

3.3.1 First Stage. Validation of Existence and Uniqueness

In the first stage, the validation of existence and uniqueness is carried out (Section 2.2.2) and an appropriate time step h_j as well as an *a priori* enclosure $[\tilde{\mathbf{y}}_j] \supseteq \mathbf{y}(t; t_j, [\mathbf{y}_j], [\boldsymbol{\theta}])$, for all $t \in [t_j, t_{j+1}]$ with $t_{j+1} = t_j + h_j$ are obtained making use of the High Order Enclosure (HOE) approach [48]. According to this approach h_j and $[\tilde{\mathbf{y}}_j]$ must satisfy the following equation:

$$[\tilde{\mathbf{y}}_j] = [\mathbf{y}_j] + \sum_{i=1}^{k-1} [0, h_j]^i \mathbf{f}^{[i]}([\mathbf{y}_j], [\boldsymbol{\theta}]) + [0, h_j]^k \mathbf{f}^{[k]}([\tilde{\mathbf{y}}_j^0], [\boldsymbol{\theta}]) \subseteq [\tilde{\mathbf{y}}_j^0] \tag{3.3}$$

where k is the order of the Taylor series expansion, $[\mathbf{y}_j]$ is the vector of tight enclosures of the solutions with ranges in $[\tilde{\mathbf{y}}_j^0]$, $[\boldsymbol{\theta}]$ is the vector of system parameters and $\mathbf{f}^{[i]}$ are the

Taylor coefficients defined according to

$$\begin{aligned}\mathbf{f}^{[0]}(\mathbf{y},\boldsymbol{\theta}) &= \mathbf{y}_j \\ \mathbf{f}^{[i]}(\mathbf{y},\boldsymbol{\theta}) &= \frac{1}{i}\left(\frac{\partial \mathbf{f}^{[i-1]}}{\partial \mathbf{y}}\mathbf{f}\right)(\mathbf{y},\boldsymbol{\theta}) \text{ for } i \geq 1\end{aligned} \tag{3.4}$$

These Taylor coefficients can be calculated using automatic differentiation (Section 2.2.1). To obtain the *a priori* enclosure $[\tilde{\mathbf{y}}_j]$ the computation starts with $[\tilde{\mathbf{y}}_j^0]$ wide enough so that it encloses $[\mathbf{y}_j]$ and an initial step size h_j. Since the problem has known initial conditions, we can always make sure to select a $[\tilde{\mathbf{y}}_0^0]$ that encloses $[\mathbf{y}_0]$. Equation (3.3) is substituted and an *a priori* enclosure $[\tilde{\mathbf{y}}_j]$ is computed. If $[\tilde{\mathbf{y}}_j] \not\subseteq [\tilde{\mathbf{y}}_j^0]$, then the step size h_j is decreased until the inclusion is satisfied.

3.3.2 Second Stage. Tightening of the Solution

The second stage involves the computation of a tight enclosure (Section 2.2.3) $[\mathbf{y}_{j+1}] \supseteq \mathbf{y}(t_{j+1}; t_0, [\mathbf{y}_0], [\boldsymbol{\theta}])$ given interval bounds $[\mathbf{y}_j]$ at t_j. This stage refines the *a priori* enclosure $[\tilde{\mathbf{y}}_j]$ on $t \in [t_j, t_{j+1}]$ provided in the first stage. It satisfies the next equation

$$\begin{aligned}[\mathbf{y}_{j+1}] = \overbrace{\hat{\mathbf{y}}_j + \sum_{i=1}^{k-1} h_j^i \mathbf{f}^{[i]}(\hat{\mathbf{y}}_j, \hat{\boldsymbol{\theta}})}^{\mathbf{u}_{j+1}} + \overbrace{\left\{\mathbf{I} + \sum_{i=1}^{k-1} h_j^i \frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{y}}([\mathbf{y}_j],[\boldsymbol{\theta}])\right\}}^{[\mathbf{S}_{j+1}^y]}([\mathbf{y}_j] - \hat{\mathbf{y}}_j) \\ + \underbrace{\left\{\sum_{i=0}^{k-1} h_j^i \frac{\partial \mathbf{f}^{[i]}}{\partial \boldsymbol{\theta}}([\mathbf{y}_j],[\boldsymbol{\theta}])\right\}}_{[\mathbf{S}_{j+1}^\theta]}([\boldsymbol{\theta}] - \hat{\boldsymbol{\theta}}) + \underbrace{h_j^k \mathbf{f}^{[k]}([\tilde{\mathbf{y}}_j],[\boldsymbol{\theta}])}_{[\mathbf{z}_{j+1}]}\end{aligned} \tag{3.5}$$

where $\mathbf{I}$ is the identity matrix and $\hat{\mathbf{y}}_j = (\underline{\mathbf{y}_j} + \overline{\mathbf{y}_j})/2$. For the sake of simplicity, we rewrite equation (A.4) as

$$[\mathbf{y}_{j+1}] = \mathbf{u}_{j+1} + [\mathbf{S}_{j+1}^y]([\mathbf{y}_j] - \hat{\mathbf{y}}_j) + [\mathbf{S}_{j+1}^\theta]([\boldsymbol{\theta}] - \hat{\boldsymbol{\theta}}) + [\mathbf{z}_{j+1}] \tag{3.6}$$

In this method, the interval matrix-vector product $[\mathbf{S}_{j+1}^y]([\mathbf{y}_j] - \hat{\mathbf{y}}_j)$ in equation (3.6) is known to be one of the main contributors of the wrapping effect [43]. Because of this, a number of methods have been developed in order to avoid direct evaluations of this matrix-

vector product [47]. In this chapter, the QR factorization technique devised by [36] is used in the interval Taylor series method. This technique or a similar variation is also used in Nedialkov [46], Lin and Stadtherr [31], Sahlodin and Chachuat [66] and Sahlodin and Chachuat [65].

The QR factorization technique consists of the substitution of the term $[\mathbf{S}^y_{j+1}]([\mathbf{y}_j] - \hat{\mathbf{y}}_j)$ by another term containing a matrix that induces an orthogonal coordinate system. Such substitution often provides a better enclosure than the original coordinate system.

$$[\mathbf{y}_{j+1}] = \mathbf{u}_{j+1} + ([\mathbf{S}^y_{j+1}]\mathbf{A}_j)[\mathbf{\Gamma}_j] + [\mathbf{S}^\theta_{j+1}]([\boldsymbol{\theta}] - \hat{\boldsymbol{\theta}}) + [\mathbf{z}_{j+1}] \tag{3.7}$$

where,

$$[\mathbf{\Gamma}_{j+1}] = \mathbf{A}^{-1}_{j+1}([\mathbf{z}_{j+1}] - \hat{\mathbf{z}}_{j+1}) + \mathbf{A}^{-1}_{j+1}([\mathbf{S}^y_{j+1}]\mathbf{A}_j)[\mathbf{\Gamma}_j] + (\mathbf{A}^{-1}_{j+1}[\mathbf{S}^\theta_{j+1}])([\boldsymbol{\theta}] - \hat{\boldsymbol{\theta}}) \tag{3.8}$$

Here, $[\mathbf{\Gamma}_0] = [\mathbf{y}_0] - \hat{\mathbf{y}}_0$, $\mathbf{A}_0 = \mathbf{I}$ and $\mathbf{A}_{j+1}$ is chosen as the orthogonal matrix in the QR factorization of $\hat{\mathbf{S}}^y_{j+1}\mathbf{A}_j$, the parallelepiped enclosure of $[\mathbf{\Gamma}_{j+1}]$. This form of rearranging the matrix-vector product was first reported by [36].

As discussed before, overestimation is present in verified integration methods because intervals and relaxations provide only approximations of the true solution set. The ITS method on its own is not able to address significant widths in the system parameters or initial conditions. When intervals with big widths are used in the method, overestimation is generated by the dependency, cancellation and wrapping effect problems. The excess of overestimation causes the bounds to be overconservative and in the worst case makes the bounds tend to $\pm\infty$ and consequently the integration stops. The next section describes the overestimation sources, some techniques to reduce it and an implementation of these techniques in a verified integrator.

3.4 Reduction of the Overestimation via Interval Contractors

Verified methods exist for the solution of IVPs for ODEs which often rely on interval analysis [42]. These methods provide upper and lower bounds in which the true solution of the problem is guaranteed to be contained. A major challenge in these methods is the reduction of the overestimation generated in the integration process, which is mainly caused by the dependency and wrapping effect problems.

To tackle this problem, some approaches have been proposed such as the ITS with Hermite-Obreschkoff approach by Nedialkov [46], the ITS with convex/concave remainder term using McCormick relaxations [66], the differential inequalities with interval analysis approach [69], the validated enclosure of an approximate solution for ODEs and DAEs by [63] and the Taylor Models approach with interval remainder term [31], with convex/concave remainder term using McCormick relaxations [65] and with ellipsoidal remainder term [24].

However, there is still a need for effective ways to reduce the overestimation in order to address problems with more state variables and bigger uncertainties. Being able to address bigger uncertain values means that in the dynamic optimisation problem, we are able to provide bounds for the objective function and gradients of larger regions of the decision space, which leads to speed-up of the algorithm as less calls to the verified integration algorithm are required. Section 3.4.1 presents the sources of overestimation in interval analysis which directly affect verified ODE integration. Interval contractors able to reduce the overestimation in nonlinear functions are presented in Section 3.4.2 as an option for tackling this problem. Finally, Section 3.4.3 describes an algorithm implementing interval contractors in an ITS method to enhance its overestimation reduction capabilities.

3.4.1 Overestimation in Verified Simulation

In verified simulation, there are three main sources of overestimation namely, dependence, cancellation and wrapping.

Dependence and Cancellation

The interval Taylor series method makes no special treatment of this source of overestimation. Every mathematical operation in the method is done using interval arithmetic. As will be seen in future sections, mechanisms for reducing the overestimation using contractors will be used to tackle this issue as a whole.

Wrapping

The techniques used by this method to address this source of overesimation are the QR factorization technique and the parallelepiped technique [36]. The method allows every enclosure to be more accurate since it induces a coordinate transformation that favours the current enclosure.

3.4.2 Fixed-point Interval Contractors

When a real valued function is evaluated using interval arithmetic, usually some overestimation is present in the range of the function. Interval contractors offer the possibility to contract the estimated range in an interval evaluated function. Some of these contractors, such as the Krawczyk contractor and the Newton contractor, are able to contract nonlinear functions. Consider the case in which we have n_y variables linked by n_f relations of the form

$$f_q(y_1, y_2, ..., y_{n_y}) = 0,\ q \in 1, 2, ..., n_f \tag{3.9}$$

Each variable y_j belongs to a domain $[y_j]$, an interval. Equation (3.9) can be written in a vector form and a constraint satisfaction problem (CSP) can be formulated as in equation (3.10).

$$(\mathbf{f}(\mathbf{y}) = \mathbf{0},\ \mathbf{y} \in [\mathbf{y}_j]) \tag{3.10}$$

The solution set B of (3.10) is defined as

$$B = \{\mathbf{y} \in [\mathbf{y}_j] | \mathbf{f}(\mathbf{y}) = \mathbf{0}\} \tag{3.11}$$

Contracting the CSP in (3.10) means replacing $[\mathbf{y}_j]$ by a smaller domain $[\mathbf{y}'_j]$ such that the solution set remains unchanged, i.e., $B \subset [\mathbf{y}'] \subset [\mathbf{y}]$. The contractors considered in this work are interval counterparts of classical point algorithms such as Gauss-Seidel and Newton algorithms. For more details see [27] and for application examples in deterministic global optimisation refer to [3].

As nonlinear problems are being considered, contractors for nonlinear functions were used in this work as opposed to contractors for linear functions (Gauss elimination, Gauss-Seidel, Linear Programming). The Krawczyk and Newton/Gauss-Seidel nonlinear contractors were implemented in the method as described in Section 3.4.3. This method was then used in case studies to assess the effectiveness in the reduction of the overestimation. Other contractors such as the forward-backward contractor can be used in nonlinear functions however it requires the definition of primitive constraints (constraints involving only elementary operations and intrinsic functions). This contractor will be topic of the next chapter.

Interval Krawczyk Contractor

The Krawczyk contractor (see Table 3.1) considers a CSP as in (3.10) where the number of variables is the same as the number of relations and $\mathbf{f}$ is assumed to be differentiable. The function

$$\boldsymbol{\psi}(\mathbf{y}) = \mathbf{y} - \mathbf{M}\mathbf{f}(\mathbf{y}) \tag{3.12}$$

is a fixed-point subsolver for (3.10) where $\mathbf{M}$ is assumed to be an invertible matrix. Its centred inclusion function is

$$\boldsymbol{\Psi}([\mathbf{y}_j]) = \boldsymbol{\psi}(\hat{\mathbf{y}}_j) + \mathbf{J}_{\psi}([\mathbf{y}_j]) \cdot ([\mathbf{y}_j] - \hat{\mathbf{y}}_j) \tag{3.13}$$

where $\mathbf{J}_{\boldsymbol{\psi}}$ is an inclusion function for the Jacobian matrix of $\boldsymbol{\psi}$ with $\hat{\mathbf{y}}_j$ as the midpoint of $[\mathbf{y}_j]$. The intersection between the original domain and the one obtained with (3.13) results in the fixed-point contractor as in equation (3.14).

$$[\mathbf{y}_j] \leftarrow [\mathbf{y}_j] \cap \{\boldsymbol{\psi}(\hat{\mathbf{y}}_j) + \mathbf{J}_{\psi}([\mathbf{y}_j]) \cdot ([\mathbf{y}_j] - \hat{\mathbf{y}}_j)\}\,. \tag{3.14}$$

Table 3.1: Krawczyk contractor algorithm

`KContractor(In:` $[\mathbf{y}_j]$, $[\mathbf{S}^y_{j+1}]$, $\mathbf{S}^y_{j+1}(\hat{\mathbf{y}}_j, \hat{\boldsymbol{\theta}})$, $\mathbf{g}(\hat{\mathbf{y}}_j, [\boldsymbol{\theta}])$; `Out:` $[\mathbf{y}_j]$)
begin
$\hat{\mathbf{y}}_j = (\underline{\mathbf{y}_j} + \overline{\mathbf{y}_j})/2$
$\mathbf{M} = (\mathbf{S}^y_{j+1}(\hat{\mathbf{y}}_j, \hat{\boldsymbol{\theta}}))^{-1}$
$[\mathbf{J}_\psi] = \mathbf{I} - \mathbf{M}[\mathbf{S}^y_{j+1}]$
$[\mathbf{r}] = \hat{\mathbf{y}}_j - \mathbf{M}\mathbf{g}(\hat{\mathbf{y}}_j, [\boldsymbol{\theta}]) + [\mathbf{J}_\psi] \cdot ([\mathbf{y}_j] - \hat{\mathbf{y}}_j)$
$[\mathbf{y}_j] \leftarrow [\mathbf{y}_j] \cap [\mathbf{r}]$
end

After substituting (3.12) into (3.14) we get

$$[\mathbf{y}_j] \leftarrow [\mathbf{y}_j] \cap \{\hat{\mathbf{y}}_j - \mathbf{M}\mathbf{f}(\hat{\mathbf{y}}_j) + (\mathbf{I} - \mathbf{M}\mathbf{J}_\mathbf{f}([\mathbf{y}_j])) \cdot ([\mathbf{y}_j] - \hat{\mathbf{y}}_j)\} \tag{3.15}$$

where $\mathbf{M}$ is taken as $(\mathbf{J}_\mathbf{f}(\hat{\mathbf{y}}_j))^{-1}$.

Interval Newton/Gauss-Seidel Contractor

We consider again a CSP as in (3.10) and apply the mean value theorem to obtain

$$\{\mathbf{f}(\hat{\mathbf{y}}_j) + \mathbf{J}_\mathbf{f}(\boldsymbol{\xi})(\mathbf{y}_j - \hat{\mathbf{y}}_j) = \mathbf{0} \mid \ \mathbf{y}_j \in [\mathbf{y}_j],\ \boldsymbol{\xi} \in [\mathbf{y}_j]\} \tag{3.16}$$

The CSP in (3.16) can be arranged as

$$\begin{pmatrix} \mathbf{A}\mathbf{p} + \mathbf{f}(\hat{\mathbf{y}}_j) = 0 \\ \mathbf{p} = (\mathbf{y}_j - \hat{\mathbf{y}}_j) \\ \mathbf{A} = \mathbf{J}_f(\boldsymbol{\xi}) \\ \mathbf{b} = -\mathbf{f}(\hat{\mathbf{y}}_j) \\ \mathbf{y}_j \in [\mathbf{y}_j],\ \boldsymbol{\xi} \in [\mathbf{y}_j] \end{pmatrix} \tag{3.17}$$

In this way, a linear contractor can be used such as the Gauss-Seidel contractor (see Table 3.2). The Gauss-Seidel contractor is able to contract domains of linear systems of the

form

$$\mathbf{A}\mathbf{p} - \mathbf{b} = \mathbf{0} \tag{3.18}$$

If $\mathbf{A}$ is square, it can be decomposed as the sum of a diagonal matrix and a matrix with zeroes on its diagonal (extdiag): $\mathbf{A} = diag(\mathbf{A}) + extdiag(\mathbf{A})$.

Now $\mathbf{A}\mathbf{p} - \mathbf{b} = \mathbf{0}$ is equivalent to

$$diag(\mathbf{A})\mathbf{p} + extdiag(\mathbf{A})\mathbf{p} = \mathbf{b}. \tag{3.19}$$

Also if $\mathbf{A}$ is invertible, then (3.19) can be rewritten as

$$\mathbf{p} = (diag(\mathbf{A}))^{-1}(\mathbf{b} - extdiag(\mathbf{A})\mathbf{p}) \tag{3.20}$$

Hence, the solution of the Gauss-Seidel contractor is defined as the intersection of the original domain $[\mathbf{p}]$ and the new $[\mathbf{p}]$ calculated with (3.20). This results in:

$$[\mathbf{p}] \leftarrow [\mathbf{p}] \cap (diag([\mathbf{A}]))^{-1}([\mathbf{b}] - extdiag([\mathbf{A}])[\mathbf{p}]) \tag{3.21}$$

Finally, the Gauss-Seidel contractor solution in (3.21) is used to update $[\mathbf{y}_j]$ and the intersection $[\mathbf{y}_j] \leftarrow [\mathbf{y}_j] \cap ([\mathbf{p}] + \hat{\mathbf{y}}_j)$ is obtained to finish with the Newton procedure (see Table 3.3).

3.4.3 Interval Contractors in the Verified Method

In order to apply the interval contractors in the verified method with the aim of reducing the overestimation, this work considers an implicit form of equation (3.6) so as to formulate a CSP ($\mathbf{f}(\mathbf{y}) = 0$) at each time step. In a similar way, if in other verified ODE solvers (e.g. based on Taylor models or differential inequalities), this formulation can be done, then interval contractors can be applied as well. If we make this reformulation, it is possible to consider the new implicit equation as a CSP problem in the form of (3.10). The formulation

Table 3.2: Gauss-Seidel Preconditioned contractor algorithm

`GSContractor`(`In`: $[\mathbf{A}]$, $[\mathbf{p}]$, $[\mathbf{b}]$; `Out`: $[\mathbf{A}]$, $[\mathbf{p}]$, $[\mathbf{b}]$)
begin
$\mathbf{A}_0 = \hat{\mathbf{A}}$
$[\mathbf{A}'] = \mathbf{A}_0[\mathbf{A}]$
$[\mathbf{b}'] = \mathbf{b}_0[\mathbf{b}]$
$diag([\mathbf{A}']) + extdiag([\mathbf{A}']) = [\mathbf{A}']$
$[\mathbf{p}] \leftarrow [\mathbf{p}] \cap (diag([\mathbf{A}']))^{-1}([\mathbf{b}'] - extdiag([\mathbf{A}'])[\mathbf{p}])$
$[\mathbf{b}] \leftarrow \mathbf{A}_0[\mathbf{b}'] \cap [\mathbf{b}]$
$[\mathbf{A}] \leftarrow \mathbf{A}_0[\mathbf{A}'] \cap [\mathbf{A}]$
end

Table 3.3: Newton/Gauss-Seidel contractor algorithm

`NGSContractor`(`In`: $[\mathbf{y}_{j+1}]$, $[\mathbf{S}^y_{j+1}]$, $\mathbf{g}(\hat{\mathbf{y}}_j, [\boldsymbol{\theta}])$; `Out`: $[\mathbf{y}_{j+1}]$)
begin
$\hat{\mathbf{y}}_j = (\underline{\mathbf{y}_j} + \overline{\mathbf{y}_j})/2$
$[\mathbf{A}] = [\mathbf{S}^y_{j+1}]$
$[\mathbf{p}] = [\mathbf{y}_j] - \hat{\mathbf{y}}_j$
$[\mathbf{p}] \supseteq [\mathbf{p}]_{GSP} =$`GSContractor`$([\mathbf{A}], [\mathbf{p}], \mathbf{g}(\hat{\mathbf{y}}_j, [\boldsymbol{\theta}]))$
$[\mathbf{y}_j] \leftarrow [\mathbf{y}_j] \cap [\mathbf{p}]$
end

is the following

$$\mathbf{g}(\mathbf{y}) = \mathbf{u}_{j+1} + ([\mathbf{S}^y_{j+1}]\mathbf{A}_j)[\mathbf{\Gamma}_j] + [\mathbf{S}^\theta_{j+1}]([\boldsymbol{\theta}] - \hat{\boldsymbol{\theta}}) + [\mathbf{z}_{j+1}] - [\mathbf{y}_{j+1}] = \mathbf{0},\ \mathbf{y} \in [\mathbf{y}_j] \tag{3.22}$$

however this reformulation does not yield a form as in (3.10) as the subtraction of identical vectors in interval arithmetic is not cancelling and the widths are added up due to the cancellation problem (Section 2.3.2). So a midpoint evaluation is performed and we get

$$\begin{aligned}\mathbf{g}(\hat{\mathbf{y}}_j) = \mathbf{u}_{j+1}(\hat{\mathbf{y}}_j, \hat{\boldsymbol{\theta}}) + \{\mathbf{S}^y_{j+1}(\hat{\mathbf{y}}_j, [\boldsymbol{\theta}])\mathbf{A}_j\}\mathbf{\Gamma}_j(\hat{\mathbf{y}}_j, [\boldsymbol{\theta}]) + \{\mathbf{S}^\theta_{j+1}(\hat{\mathbf{y}}_j, [\boldsymbol{\theta}])\}([\boldsymbol{\theta}] - \hat{\boldsymbol{\theta}}) \\ + \mathbf{z}_{j+1}(\tilde{\mathbf{y}}_j, [\boldsymbol{\theta}]) - \hat{\mathbf{y}}_{j+1}([\mathbf{y}_j], [\boldsymbol{\theta}]) = \mathbf{0}\end{aligned} \tag{3.23}$$

which has the form of (3.10). Since

$$\hat{\mathbf{y}}_{j+1}([\mathbf{y}_j], [\boldsymbol{\theta}]) = \mathbf{u}_{j+1}(\hat{\mathbf{y}}_j, \hat{\boldsymbol{\theta}}) + \hat{\mathbf{z}}_{j+1}([\tilde{\mathbf{y}}_j], [\boldsymbol{\theta}])$$

we have

$$\begin{aligned}\mathbf{g}(\hat{\mathbf{y}}_j) &= \mathbf{u}_{j+1}(\hat{\mathbf{y}}_j, \hat{\boldsymbol{\theta}}) + \{\mathbf{S}^y_{j+1}(\hat{\mathbf{y}}_j, [\boldsymbol{\theta}])\mathbf{A}_j\}\mathbf{\Gamma}_j(\hat{\mathbf{y}}_j, [\boldsymbol{\theta}]) + \{\mathbf{S}^\theta_{j+1}(\hat{\mathbf{y}}_j, [\boldsymbol{\theta}])\}([\boldsymbol{\theta}] - \hat{\boldsymbol{\theta}}) \\ &\quad + \mathbf{z}_{j+1}(\tilde{\mathbf{y}}_j, [\boldsymbol{\theta}]) - (\mathbf{u}_{j+1}(\hat{\mathbf{y}}_j, \hat{\boldsymbol{\theta}}) + \hat{\mathbf{z}}_{j+1}([\tilde{\mathbf{y}}_j], [\boldsymbol{\theta}])) = \mathbf{0} \\ &= \{\mathbf{S}^y_{j+1}(\hat{\mathbf{y}}_j, [\boldsymbol{\theta}])\mathbf{A}_j\}\mathbf{\Gamma}_j(\hat{\mathbf{y}}_j, [\boldsymbol{\theta}]) + \{\mathbf{S}^\theta_{j+1}(\hat{\mathbf{y}}_j, [\boldsymbol{\theta}])\}([\boldsymbol{\theta}] - \hat{\boldsymbol{\theta}}) \\ &\quad + \mathbf{z}_{j+1}(\tilde{\mathbf{y}}_j, [\boldsymbol{\theta}]) - \hat{\mathbf{z}}_{j+1}([\tilde{\mathbf{y}}_j], [\boldsymbol{\theta}]) = \mathbf{0}.\end{aligned}$$

Also recalling equation (3.8)

$$[\mathbf{\Gamma}_{j+1}] = \mathbf{A}^{-1}_{j+1}([\mathbf{z}_{j+1}] - \hat{\mathbf{z}}_{j+1}) + \mathbf{A}^{-1}_{j+1}([\mathbf{S}^y_{j+1}]\mathbf{A}_j)[\mathbf{\Gamma}_j] + (\mathbf{A}^{-1}_{j+1}[\mathbf{S}^\theta_{j+1}])([\boldsymbol{\theta}] - \hat{\boldsymbol{\theta}})$$

we get

$$\mathbf{g}(\hat{\mathbf{y}}_j) = \mathbf{A}_{j+1}\mathbf{\Gamma}_{j+1}(\hat{\mathbf{y}}_j, [\boldsymbol{\theta}]) \tag{3.24}$$

which corresponds to the global error in the verified simulation [36].

When there is no uncertainty then $\mathbf{g}(\hat{\mathbf{y}}_j) = \mathbf{A}_{j+1}\mathbf{\Gamma}_{j+1}(\hat{\mathbf{y}}_j, [\boldsymbol{\theta}]) = \mathbf{0}$. In the present work,

uncertainty has been considered in the initial conditions and parameters of the ODEs; it will be shown in the next section that when a number of iterations of the Krawczyk and Newton steps are used in Equation (3.24), reduction of the overestimation is achieved.

In summary, since Equation (3.24) is defined at each time step, it is possible to implement the contractors as in an equation of the form of (3.10). In this way, at each time step, the interval Taylor series method obtains an interval ($[\mathbf{y}_{j+1}]$) that encloses the solution of the problem. The midpoint $\hat{\mathbf{y}}_{j+1}$ is then used to define equation (3.23). After a number of contractor iterations, an interval vector for the current time step is obtained. The resulting interval vector has a width that is smaller or equal than the original width of the input interval vector. The steps to obtain the guaranteed enclosures using contractors are illustrated in Table 3.4.

3.4.4 Implementation and Third Party Libraries

The methods explained before were implemented in C++ and third party libraries were used for defining the interval type and for performing automatic differentiation. In particular the library PROFIL/BIAS was used for defining the interval type and FADBAD++ for the automatic differentiation.

PROFIL/BIAS (Programmer's Runtime Optimized Fast Interval Library/Basic Interval Arithmetic Subroutines) is an open source C++ library (http://www.ti3.tuhh.de/keil/profil/index_e.html) to define an interval type that consists of numbers defined by a lower and an upper bound. The library also defines the interval arithmetic operations and interval intrinsic functions for scalar, vectors and matrices.

PROFIL/BIAS defines the type so that every single operation can be treated as an interval. However, every amount needs to be defined as an interval. If a point valued amount needs to be used, then it is defined as a degenerate interval. For example, 0 is defined as [0,0] and 1 as [1,1] and the vector $\{0, 0\}$ is defined as $\{[0, 0], [0, 0]\}$. This library does not support division by 0 so it throws an undefined behaviour and the program has to stop.

Table 3.4: Interval Taylor series with fixed-point contractors algorithm

Initialise: $\mathbf{A}_0 = \mathbf{I}$, $[\mathbf{\Gamma}_0] = [\mathbf{y}_0] - \hat{\mathbf{y}}_0$

`ITSContractor`(`In`: $\mathbf{A}_j$, $[\mathbf{\Gamma}_j]$, h_j, $[\tilde{\mathbf{y}}_j]$, $[\mathbf{y}_j]$, $[\boldsymbol{\theta}]$; `Out`: $\mathbf{A}_{j+1}$, $[\mathbf{\Gamma}_{j+1}]$, $[\mathbf{y}_{j+1}]$)

begin

$\mathbf{u}_{j+1} = \hat{\mathbf{y}}_j + \sum_{i=1}^{k-1} h_j^i \mathbf{f}^{[i]}(\hat{\mathbf{y}}_j, \hat{\boldsymbol{\theta}})$

$[\mathbf{z}_{j+1}] = h_j^k \mathbf{f}^{[k]}([\tilde{\mathbf{y}}_j], [\boldsymbol{\theta}])$

$[\mathbf{S}_{j+1}^y] = \mathbf{I} + \sum_{i=1}^{k-1} h_j^i \frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{y}}([\mathbf{y}_j], [\boldsymbol{\theta}])$

$[\mathbf{S}_{j+1}^\theta] = \sum_{i=0}^{k-1} h_j^i \frac{\partial \mathbf{f}^{[i]}}{\partial \boldsymbol{\theta}}([\mathbf{y}_j], [\boldsymbol{\theta}])$

$\mathbf{Q}_j \mathbf{R}_j = \hat{\mathbf{S}}_{j+1}^y \mathbf{A}_j$

$\mathbf{A}_{j+1} = \mathbf{Q}_j$

$[\mathbf{\Gamma}_{j+1}] = \mathbf{A}_{j+1}^{-1}([\mathbf{z}_{j+1}] - \hat{\mathbf{z}}_{j+1}) + \mathbf{A}_{j+1}^{-1}([\mathbf{S}_{j+1}^y]\mathbf{A}_j)[\mathbf{\Gamma}_j] + (\mathbf{A}_{j+1}^{-1}[\mathbf{S}_{j+1}^\theta])([\boldsymbol{\theta}] - \hat{\boldsymbol{\theta}})$

$[\mathbf{y}_{j+1}] = \mathbf{u}_{j+1} + ([\mathbf{S}_{j+1}^y]\mathbf{A}_j)[\mathbf{\Gamma}_j] + [\mathbf{S}_{j+1}^\theta]([\boldsymbol{\theta}] - \hat{\boldsymbol{\theta}}) + [\mathbf{z}_{j+1}]$

Newton/Gauss-Seidel contractor:

$\mathbf{g}(\hat{\mathbf{y}}_j, [\boldsymbol{\theta}]) = \mathbf{A}_{j+1}[\mathbf{\Gamma}_{j+1}](\hat{\mathbf{y}}_j, [\boldsymbol{\theta}])$

$[\mathbf{y}_{j+1}] \supseteq [\mathbf{y}_{j+1}]_N =$`NGSContractor`$([\mathbf{y}_{j+1}], [\mathbf{S}_{j+1}^y], \mathbf{g}(\hat{\mathbf{y}}_j, [\boldsymbol{\theta}]))$

or

$[\mathbf{y}_{j+1}] \supseteq [\mathbf{y}_{j+1}]_N =$`KContractor`$([\mathbf{y}_{j+1}], [\mathbf{S}_{j+1}^y], \mathbf{g}(\hat{\mathbf{y}}_j, [\boldsymbol{\theta}]))$

$[\mathbf{y}_{j+1}] \leftarrow [\mathbf{y}_{j+1}] \cap [\mathbf{y}_{j+1}]_N$

end

FADBAD++ (Forward Automatic Differentiation, Backward Automatic Differentiation) is an open source C++ library (http://www.imm.dtu.dk/∼kajm/FADBAD/) to perform automatic differentiation. Specifically, the library allows to compute backward automatic differentiation, forward automatic differentiation and Taylor series expansion. The library can be overloaded with types other than doubles, for example, intervals from PROFIL/BIAS can be used to obtain interval derivatives.

Figure 3.1 represents the program used to implement an ITS method with interval contractors NGS and K. In the top of the figure (file `funcivp.cpp`) the mathematical model of the initial value problem is defined, the Taylor coefficients ($\mathbf{f}^{[i]}([\mathbf{y}_j], [\boldsymbol{\theta}])$) and the Jacobian $[\mathbf{S}^y_{j+1}]$ are computed next in `tcfunc.cpp` and `jacfunc.cpp`. The information obtained in these programs is enough to compute the first and second stages which are the high order enclosure (`HOE.cpp`) and the tighter enclosure (`ITS.cpp`). The information from `jacfunc.cpp` is also input to the contractor routines Newton/Gauss-Seidel and Krawczyk, `Ncontractor.cpp` and `Kcontractor.cpp`, respectively. Finally, `profiles.cpp` is in charge of iterating the program and arranging the information of the trajectories.

3.5 Numerical Case Studies

Numerical experiments on test problems from chemical and biochemical processes were carried out. Seven ODE models were used in the experiments and in each model, the interval contractors were applied alongside an interval Taylor series method to demonstrate the effectiveness of the methods. The first six problems are traditional Chemical Engineering problems. The seventh is a model describing the kinetics of the activity of the glucagon hormone, which is central to the regulation of glucose in the blood stream. To prevent hyper- or hypo-glycaemia glucose must be kept within upper and lower bounds but need not be controlled to a set point since it will vary with meals and normal metabolic activity. The model is part of a larger model of liver function [23, 78]. Table 3.5 shows a summary of the test problems used, which include columns with the number of state variables, the number of system parameters and the importance in dynamic optimisation. Each of the test

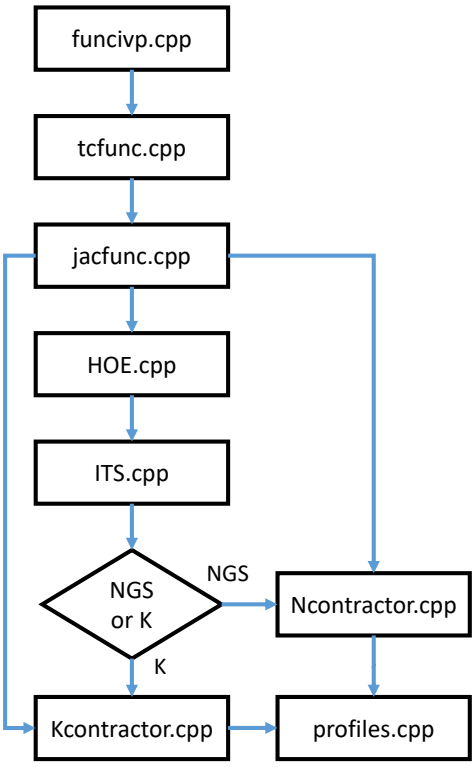


Figure 3.1: Diagram of the C++ programs to implement an ITS method with contractors

problems in the table is then extended in the subsequent subsections. The mathematical model, a table with the uncertain parameters, the graphs of two of the states and a table of results are given in each subsection.

Each test problem subsection also presents the condition number of the problem across the uncertain parameters and time horizon considered. The condition number gives a measure of how sensitive the problem output is when its inputs are perturbed. Problems with small condition numbers are well-conditioned and so small changes in the inputs yield small changes in the outputs. On the other hand, problems with a large condition number are ill-conditioned, which means that small changes in the inputs can produce arbitrarily large changes in the outputs. Preliminary, one can expect that problems with large condition to be more challenging when trying to compute upper and lower bounds. The condition number of a differentiable function f can be calculated by:

$$\kappa = \frac{\|J(x)\|}{\|f(x)\|\|x\|} \tag{3.25}$$

where J is the Jacobian of f.

Results on the implementation of the Krawczyk and Newton/Gauss-Seidel (K and NGS, respectively) interval contractors in an interval Taylor series method are presented for the test problems in Tables 3.7 to 3.19. For comparison purposes, the simulations are also carried out using no contractors (NC method) and using Taylor model bounds provided by the software package VSPODE version 1.4 [34] (VSPODE method). Typically, when the NC method is used, the overestimation quickly increases and the bounds become overconservative tending towards $\pm\infty$. However, the NC method takes less time than the methods with contractors. In order to provide meaningful comparisons, the uncertain amounts to be used are subdivided, simulations are carried out for each subinterval and the results are intersected.

For example, suppose a problem using an interval of [1,2] takes 4 seconds to be simulated with one of the contractor methods and 2.5 seconds with the NC method. In the case of the NC method, the simulation stops halfway because of overestimation. The initial interval is then bisected and two simulations are then carried out with the NC method using the bisected subintervals [1,1.5] and [1.5,2] instead. Each simulation takes 2.5 seconds and are completed up to the final time horizon. The results from the two simulations are then intersected to have the bounds for the initial interval [1,2] and the times added up (5 seconds).

In the first five test problems, the number of subintervals is chosen so as to roughly equalise the time taken by the VSPODE method. In the last two case studies, the time taken by the methods with contractors without subdivision was already higher than the VSPODE method, so only the NC method was subdivided. In Tables 3.7 to 3.19 the fourth column indicates the number of initial boxes (interval vectors) used to simulate the case study.

Tables 3.7 to 3.19 report the results obtained after using the four methods on the seven test problems. The results include the widths of the bounds and the CPU time in seconds for each of the problems and the methods. The widths given corresponds to a selected time

Table 3.5: Test problems used, number of state variables and parameters and relevance in dynamic optimisation

Problem	States	Parameters	Relevance
1. First order irreversible series reaction	2	2	Parameter estimation for model development
2. First order reversible series reaction	2	4	Parameter estimation for model development
3. Exothermic batch reactor	2	8	Guaranteed safe operation
4. Two-state bioreactor	2	6	Parameter estimation for model development
5. Three-state bioreactor	3	8	Parameter estimation for model development
6. Reactor separator model	6	9	Optimal control
7. Glucagon receptor model	5	22	Guaranteed safe operation

t_s and the CPU time reported here is using an Intel$^{\mathrm{R}}$ CORE$^{\mathrm{TM}}$ i5 computer with 8Gb RAM, running Ubuntu. Depending on the problem studied, a number of contractor iterations (or Contractors as specified in Tables 3.7 to 3.19) has been used to *contract* the width of the bounds.

Figures 3.3 to 3.22 show the trajectories in the NC, K, NGS and VSPODE methods with dot-dot-dashed blue, dot-dashed orange, dashed black and solid red lines, respectively. A Monte Carlo (MC) simulation that provides an approximation of the reachable set is shown as a shaded grey area.

The algorithm was implemented in C++ and the libraries FADBAD++ [4] and PROFIL/BIAS [29] were used for the automatic differentiation and interval operations, respectively. The PROFIL/BIAS library was used since it is faster when only interval arithmetic operations and the square function are needed [84] as in the examples considered.

Table 3.6: Initial conditions and system parameters for the first order irreversible series reaction problem

Initial conditions and system parameters
$C_A(0) = 1$
$C_B(0) = 0$
Uncertain values
$k_1 = [4.5, 5.5]$
$k_2 = [0.2, 1.8]$

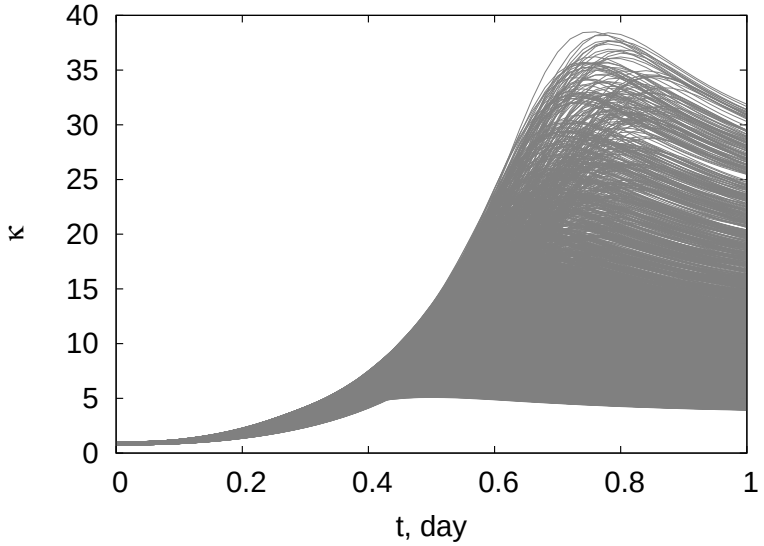


Figure 3.2: Condition number of first order irreversible series reaction across the time horizon and the parametric uncertainties

3.5.1 First Order Irreversible Series Reaction

In this section, the equations, uncertain parameters, table of results and graphs of the states are presented. The first order irreversible series reaction was solved with one level of uncertainty presented in Table 3.6. Figure 3.2 introduces the condition number of the problem which turns out to be low in this problem for the level of uncertainty considered. The effects of both uncertain parameters are shown in the graph. The separate effects have also been obtained (not reported) and they have the same order of magnitude.

$$\begin{aligned} \dot{C}_A &= -k_1 C_A \\ \dot{C}_B &= k_1 C_A - k_2 C_B \end{aligned} \tag{3.26}$$

Figures 3.3 and 3.4 show the trajectories of both states of the first order irreversible batch

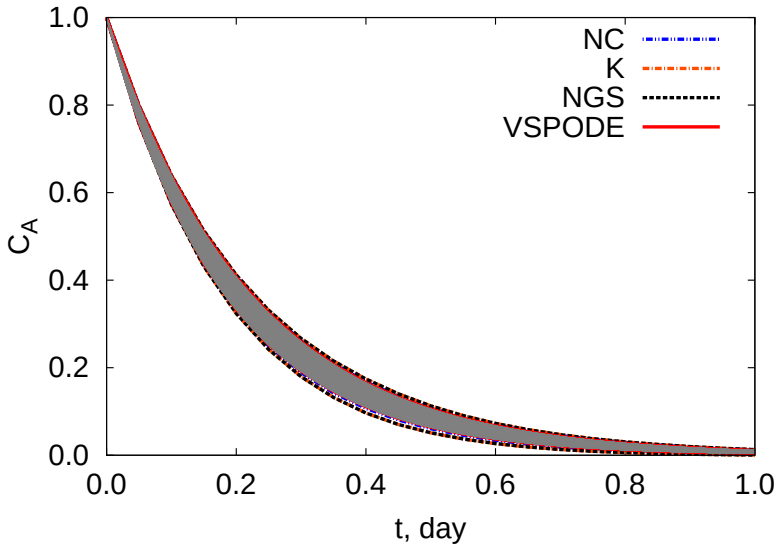


Figure 3.3: State variable C_A of first order irreversible series reaction

Table 3.7: Results on the implementation of interval contractors in the first order irreversible series reaction

Method	**Contractors**	**CPU(s)**	**Boxes**	**Width at $t_s = 1$ day**
NC	0	0.0058	4	$w(C_A(t_s))$=0.009225 $w(C_B(t_s))$=0.9190
K	1	0.0055	2	$w(C_A(t_s))$=0.01242 $w(C_B(t_s))$=1.0002108
NGS	1	0.0056	2	$w(C_A(t_s))$=0.01221 $w(C_B(t_s))$=0.9962
VSPODE	N/A	0.0053	1	$w(C_A(t_s))$=0.007070 $w(C_B(t_s))$=0.6317

reactor. The width at final time and the computational time are reported in Table 3.7. For this test problem, all the methods considered successfully bounded the trajectories within roughly the same amount of time. Furthermore, in the state C_B, the more conservative bounds were provided by the methods with contractors, which can be considered the maximum width obtained (100%), followed by the NC method with 92% of the maximum width and being the VSPODE method the tighter bounds with 63% of the maximum width obtained.

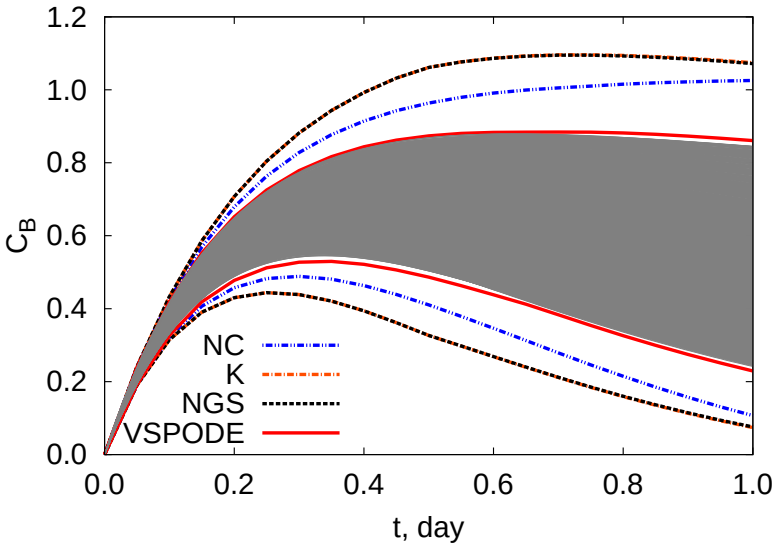


Figure 3.4: State variable C_B of first order irreversible series reaction

Table 3.8: Initial conditions and system parameters for the first order reversible series reaction problem

Initial conditions and system parameters
$C_A(0) = 1$
$C_B(0) = 0$
$k_{-1} = 2$
$k_{-2} = 20$
Uncertain values
$k_1 = [2, 6]$
$k_{-1} = [1, 3]$

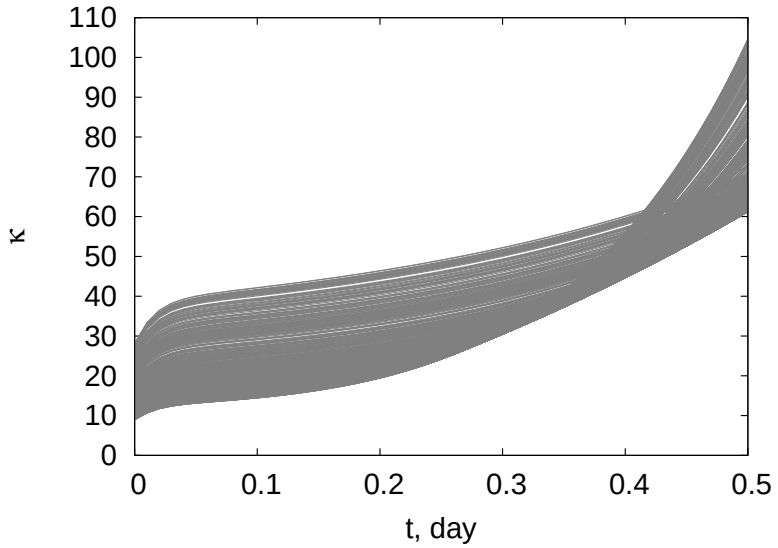


Figure 3.5: Condition number of first order reversible series reaction across the time horizon and the parametric uncertainties

3.5.2 First Order Reversible Series Reaction

The mathematical model of the problem is given in (3.27) and Table 3.8 contains the point-valued parameters as well as the uncertain parameters. Graphs with the two states of the problem are shown in 3.6 and 3.7. Table 3.9 reports the results for this test problem. The problem is considered to be well-conditioned since the condition number in this problem for this level of uncertainty is not very high according to Figure 3.5.

$$\begin{aligned} \dot{C}_A &= -k_1 C_A + k_{-1} C_B \\ \dot{C}_B &= k_1 C_A - (k_{-1} + k_2) C_B + k_{-2}(1 - C_A - C_B) \end{aligned} \tag{3.27}$$

In Figures 3.6 and 3.7, the trajectories from the four methods are displayed and in Table 3.9.

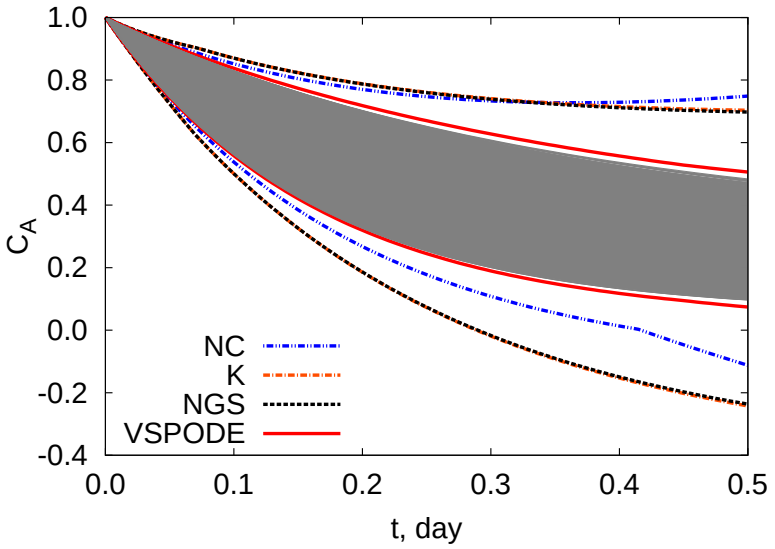


Figure 3.6: State variable C_A of first order reversible series reaction

Table 3.9: Results on the implementation of interval contractors in the first order reversible series reaction

Method	**Contractors**	**CPU(s)**	**Boxes**	**Width at $t_s = 0.5$ day**
NC	0	0.027	2	$w(C_A(t_s))$=0.8611 $w(C_B(t_s))$=0.3028
K	1	0.029	1	$w(C_A(t_s))$=0.9458 $w(C_B(t_s))$=0.5412
NGS	1	0.028	1	$w(C_A(t_s))$=0.9342 $w(C_B(t_s))$=0.5391
VSPODE	N/A	0.029	1	$w(C_A(t_s))$=0.4317 $w(C_B(t_s))$=0.1435

In this test problem, the least tight bounds were obtained by the methods with contractors. Their widths will be taken as the maximum width and will represent 100%. The NC method is again the second in tightness with a 91% for C_A and 56% for C_B of the maximum width and following VSPODE with 46% and 27% for C_A and C_B, respectively.

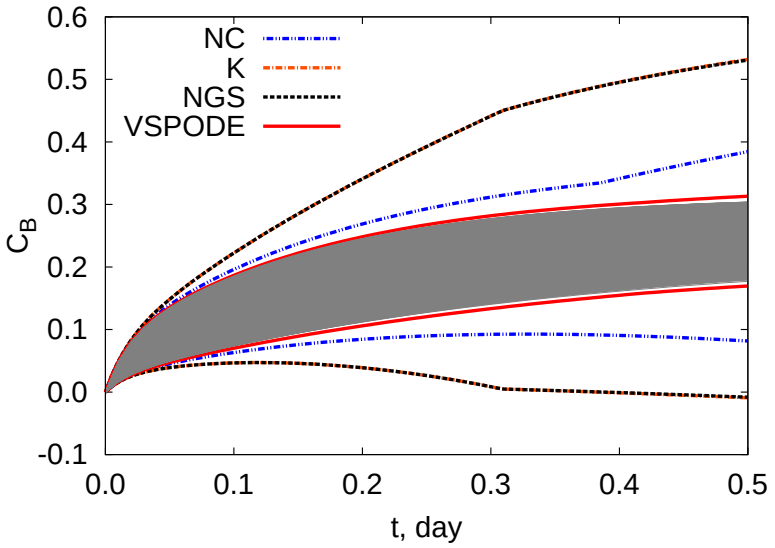


Figure 3.7: State variable C_B of first order reversible series reaction

Table 3.10: Initial conditions and system parameters for the exothermic batch reactor problem

Initial conditions and system parameters
$x(0) = 0$
$k_0 = 0.022$
$V = 0.1$
$C_p = 60$
$E_a = 6000$
$R = 8.314$
$\Delta H_R = -140000$
$UA = 3$
$C_{A0} = 10$
Uncertain values
$T(0) = [310, 410]$
$T_a = [290, 310]$

3.5.3 Exothermic Batch Reaction

This test problem includes the exponential function in its mathematical model (3.28). In Figure 3.8 it can be seen that the problem is ill-conditioned almost at the end of the final time horizon. The condition numbers using the two uncertain amounts independently were also obtained (not reported) and they both have a similar order of magnitude at the close to the end of the time horizon. The parameters of the exothermic batch reactor can be seen in Table 3.10.

$$\begin{aligned} \dot{x} &= k_0(1-x)\mathrm{e}^{-\frac{E_a}{RT}} \\ \dot{T} &= \frac{UA}{C_{A0}VC_p}(T_a - T) - \frac{\Delta H_R}{C_p}k_0(1-x)\mathrm{e}^{-\frac{E_a}{RT}} \end{aligned} \tag{3.28}$$

Table 3.11 reports the results for the simulation of the exothermic batch reactor using the four methods. The table shows the method used, the number of contractor iterations, the CPU time, the number subdivisions used in the interval amounts and the width at $t_s = 60s$. In Figures 3.9 and 3.10, the trajectories using the four methods are shown. In terms of width, the most conservative method was the NC method (100% width), followed by the other three methods which had tighter similar widths with 81% of the maximum width in x and 76% in T.

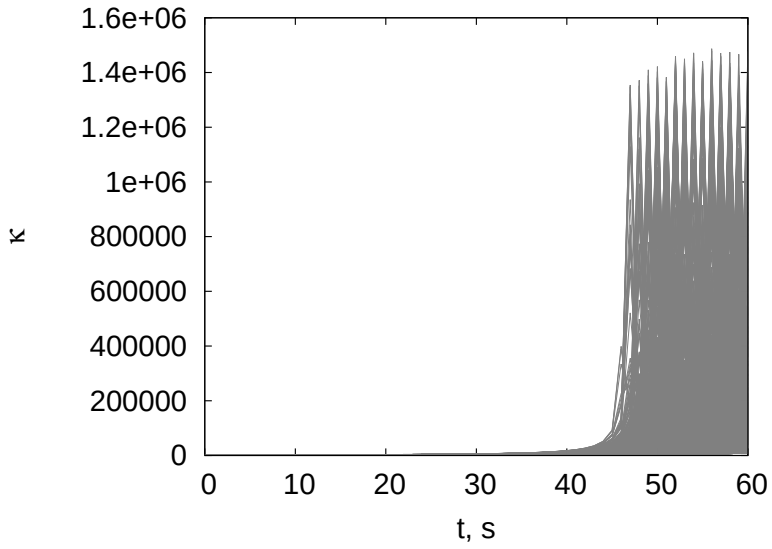


Figure 3.8: Condition number of exothermic batch reactor across the time horizon and the parametric uncertainties

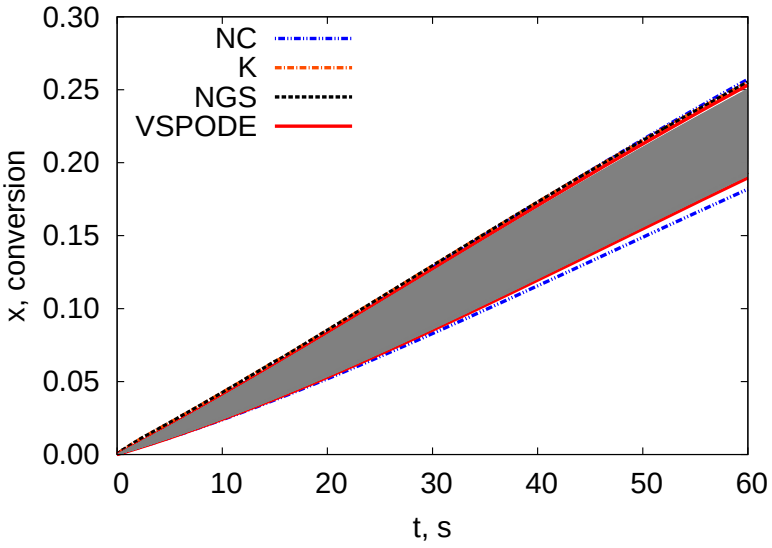


Figure 3.9: State variable x of exothermic batch reactor

Table 3.11: Results on the implementation of interval contractors in the exothermic batch reactor

Method	**Contractors**	**CPU(s)**	**Boxes**	**Width at $t_s = 60$ s**
NC	0	0.067	2	$w(x(t_s))$=0.07566 $w(T(t_s))$=75.2560
K	2	0.069	1	$w(x(t_s))$=0.06179 $w(T(t_s))$=57.3130
NGS	2	0.068	1	$w(x(t_s))$=0.06176 $w(T(t_s))$=57.1690
VSPODE	N/A	0.075	1	$w(x(t_s))$=0.06351 $w(T(t_s))$=57.6620

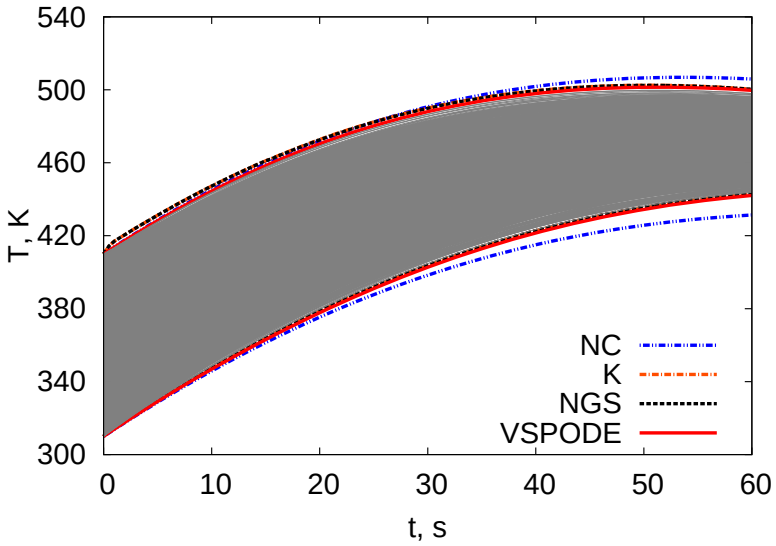


Figure 3.10: State variable T of exothermic batch reactor

Table 3.12: Initial conditions and system parameters for the two-state bioreactor problem

Initial conditions and system parameters
$S(0) = 0.8$
$\alpha = 0.5$
$k = 10.53$
$D = 0.36$
$S_f = 5.7$
$K_s = 7.0$
$k_0 = 0.022$
Uncertain values
$X(0) = [0.80, 0.85]$
$\mu_{\max} = [1.1, 1.2]$

3.5.4 Two-state Bioreactor

A two-state bioreactor is the fourth test problem. Its mathematical model can be found in Equations (3.29) and its certain and uncertain amounts are presented in Table 3.12. Also as in the previous cases, the condition number (Figure 3.11) has been obtained across the whole time horizon and the uncertain amounts. The condition number in this case is large around $t = 5$ days and $t = 10$ days.

$$\begin{aligned} \dot{X} &= (\mu - \alpha D)X \\ \dot{S} &= D(S_f - S) - k\mu X \\ \mu &= \frac{\mu_{\max} S}{K_s + S} \end{aligned} \tag{3.29}$$

Figures 3.12 and 3.13 display the trajectories of the four methods for the two states of the test problem. The figures show that in this case, the NC method did not complete the whole simulation even after subdivision of the uncertain amounts. The rest of the methods provided bounds for the whole time horizon. The methods using contractors turned out to be less tight than the VSPODE method. Considering the width of the contractor methods at $t_s = 10$ day to be the 100%, then the widths of the VSPODE method represent a 10% and a 21% of the widths of the contractor methods in X and S, respectively.

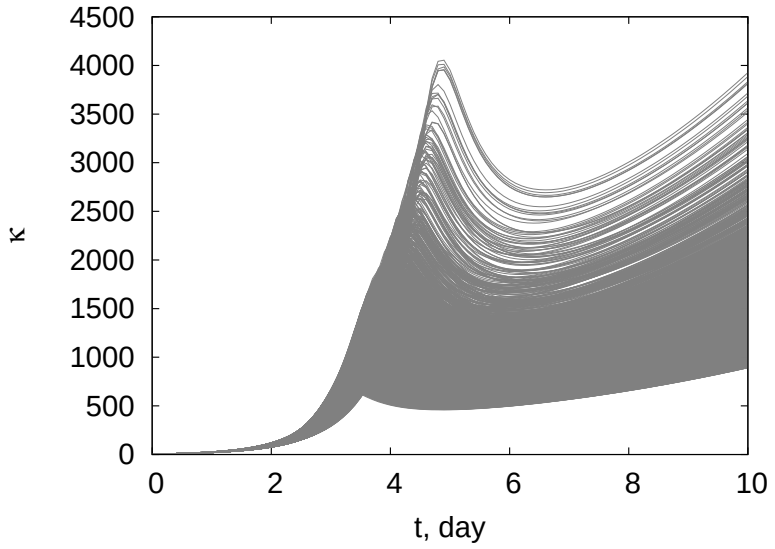


Figure 3.11: Condition number of two-state bioreactor across the time horizon and the parametric uncertainties

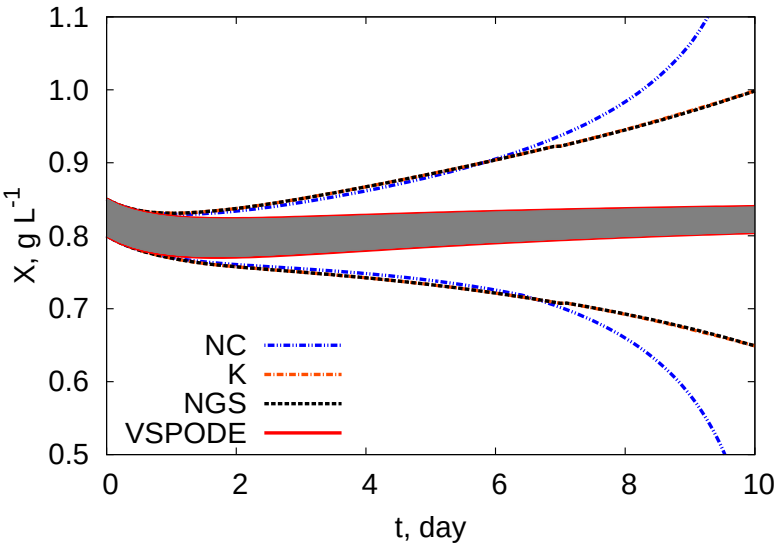


Figure 3.12: State variable X of two-state bioreactor

Table 3.13: Results on the implementation of interval contractors in the two-state bioreactor

Method	**Contractors**	**CPU(s)**	**Boxes**	**Width at $t_s = 10$ day**
NC	0	0.103	4	$w(X(t_s))$=0.9387 $w(S(t_s))$=2.1959
K	2	0.105	2	$w(X(t_s))$=0.3498 $w(S(t_s))$=0.7629
NGS	2	0.106	2	$w(X(t_s))$=0.3491 $w(S(t_s))$=0.7412
VSPODE	N/A	0.118	1	$w(X(t_s))$=0.03555 $w(S(t_s))$=0.1542

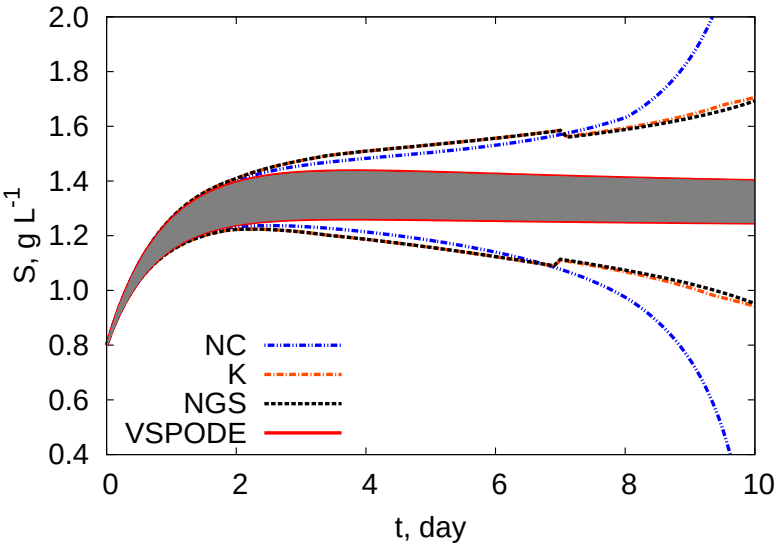


Figure 3.13: State variable S of two-state bioreactor

Table 3.14: Initial conditions and system parameters for the three-state bioreactor problem

Initial conditions and system parameters
$x_2(0) = 5.0$
$x_3(0) = 15.0$
$D = 0.202$, $x_{2f} = 20$
$Y = 0.4$
$\beta = 0.2$
$x_{3m} = 50$
$\alpha = 0.5$
Uncertain values
$x_1(0) = [6.45, 6.55]$
$\mu_{\max} = [0.46, 0.47]$
$k_s = [1.05, 1.1]$

3.5.5 Three-state Bioreactor

The present test problem increases the dimensionality considering a mathematical model of three states which can be seen in Equations (3.30). The parameters of the problems are presented in Table 3.14. In this case, the condition number (Figure 3.14) is not very large but is always increasing.

$$\begin{aligned} \dot{x}_1 &= (\mu - D)x_1 \\ \dot{x}_2 &= D(x_{2f} - x_2) - \frac{\mu x_1}{Y} \\ \dot{x}_3 &= Dx_3 + (\alpha\mu + \beta)x_1 \\ \mu &= \frac{\mu_{\max}[1 - (\frac{x_3}{x_{3m}})]x_2}{k_s + x_2} \end{aligned} \tag{3.30}$$

The results (Figures 3.15 and 3.16) indicate that the NC method is not able to bound the whole time horizon even with subdivision of the uncertain amounts. On the other hand, the methods with contractors and the VSPODE method provided bounds for the time horizon. According to Table 3.15, the most conservative methods were the methods using contractors (100% width). The VSPODE method managed to have 18% and 34% of the width of the methods using contractors for x_1 and x_2, respectively.

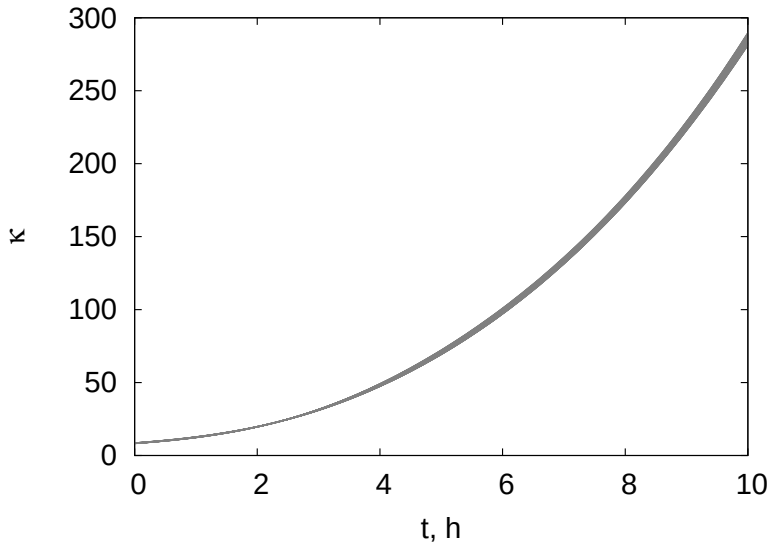


Figure 3.14: Condition number of three-state bioreactor across the time horizon and the parametric uncertainties

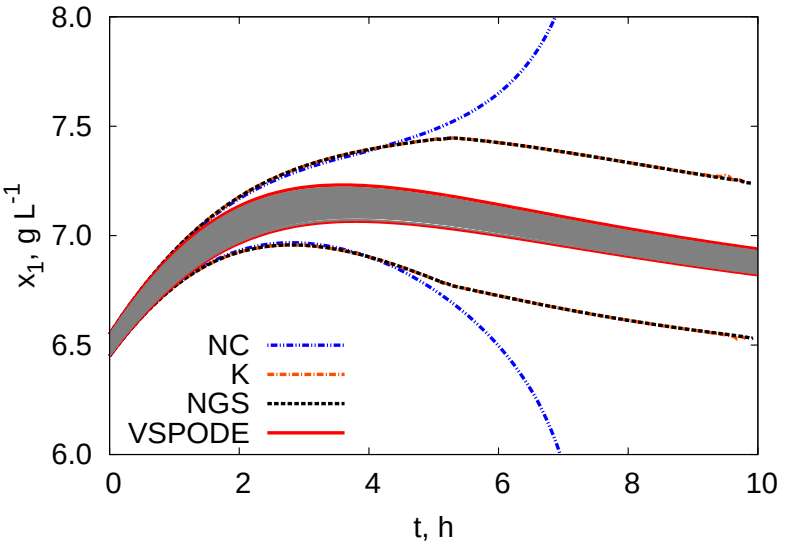


Figure 3.15: State variable x_1 of three-state bioreactor

Table 3.15: Results on the implementation of interval contractors in the three-state bioreactor

Method	**Contractors**	**CPU(s)**	**Boxes**	**Width at $t_s = 7.7$ day**
NC	0	0.669	16	$w(x_1(t_s))$=4.4063 $w(x_2(t_s))$=4.1199
K	2	0.701	2	$w(x_1(t_s))$=0.7211 $w(x_2(t_s))$=0.7876
NGS	2	0.702	2	$w(x_1(t_s))$=0.7209 $w(x_2(t_s))$=0.7868
VSPODE	N/A	0.601	1	$w(x_1(t_s))$=0.1278 $w(x_2(t_s))$=0.2666

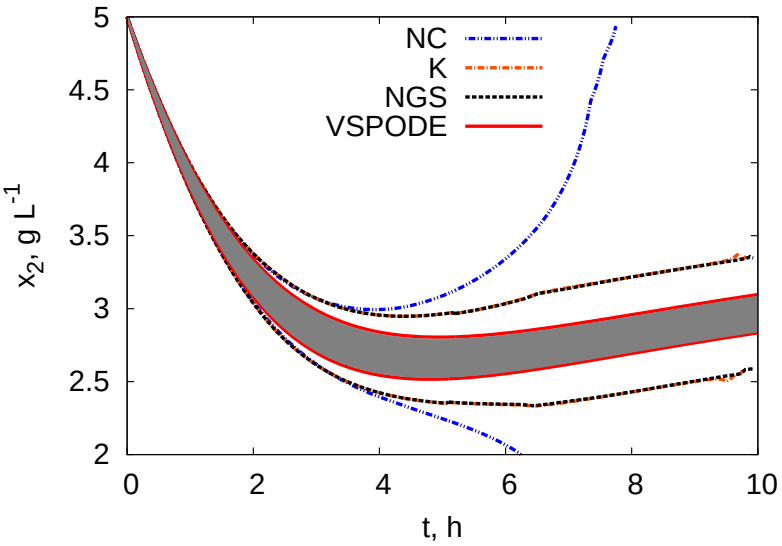


Figure 3.16: State variable x_2 of three-state bioreactor

Table 3.16: Initial conditions and system parameters for the reactor separator model problem

Initial conditions and system parameters
$x_1(0) = 0.5$
$x_2(0) = 0.0$
$x_3(0) = 0.0$
$x_4(0) = 0.0$
$k = 0.06$
$x_{F0} = 0$
$D = 1$
$H = 63.33$
$\alpha = 7.5$
$B = 1.2$
$L = 1.704$
$F = 1.0$
Uncertain values
$x_5(0) = [0.0, 0.08]$
$x_6(0) = [0.0, 0.16]$

3.5.6 Reactor separator model

Following in a similar fashion, the reactor separator model increases the number of differential equations to six and is represented by Equations (3.31). The parameters are reported in Table 3.16 and the condition number is shown in Figure 3.17. According to the figure, this number has a sudden increase early in the integration and then keeps increasing steadily.

$$\begin{aligned}
\dot{x_1} &= \frac{F+B}{H}(x_F - x_1) + kx_1(1-x_1) \\
\dot{x_2} &= (L+F+B)x_3 - Bx_2 - Vy_2 \\
\dot{x_3} &= (L+F+B)(x_4 - x_3) + V(y_2 - y_3) \\
\dot{x_4} &= (F+B)x_1 + Lx_5 - (L+F+B)x_4 + V(y_3 - y_4) \\
\dot{x_5} &= L(x_6 - x_5) + V(y_4 - y_5) \\
\dot{x_6} &= -(L+D)x_6 + Vy_5 \\
x_F &= \frac{Fx_{F0} + Bx_2}{F+B} \\
y_i &= \frac{\alpha x_i}{1 + (\alpha - 1)x_i} \quad i = 2 \ldots 5
\end{aligned} \tag{3.31}$$

Figures 3.18 and 3.19 show that all of the methods managed to provide bounds for the whole time horizon. These figures show very similar widths for all methods, so the main

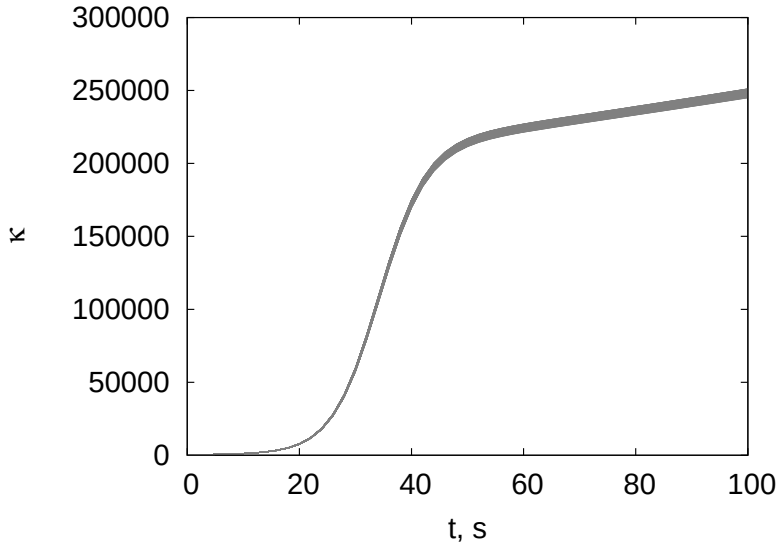


Figure 3.17: Condition number of reactor separator model across the time horizon and the parametric uncertainties

Table 3.17: Results on the implementation of interval contractors in the reactor separator model

Method	**Contractors**	**CPU(s)**	**Boxes**	**Width at $t_s = 100$ s**
NC	0	6381.810	128	$w(x_1(t_s))$=0.000238 $w(x_6(t_s))$=0.00007
K	3	79.597	1	$w(x_1(t_s))$=0.000231 $w(x_6(t_s))$=0.000069
NGS	3	82.627	1	$w(x_1(t_s))$=0.000231 $w(x_6(t_s))$=0.000069
VSPODE	0	25.875	1	$w(x_1(t_s))$=0.000229 $w(x_6(t_s))$=0.000067

difference here is the computational time. Since the methods using contractors took more time than the VSPODE method, a single box was used in these cases and 128 boxes were used in the NC method to achieve a similar level of tightness in the bounds (Table 3.17). Here the percentage will be discussed in terms of time and the NC method will be used as the 100% reference since it spent more time to perform the simulation. The K, NGS and VSPODE methods took 1.2%, 1.3% and 0.4% of the total time of the NC method, respectively.

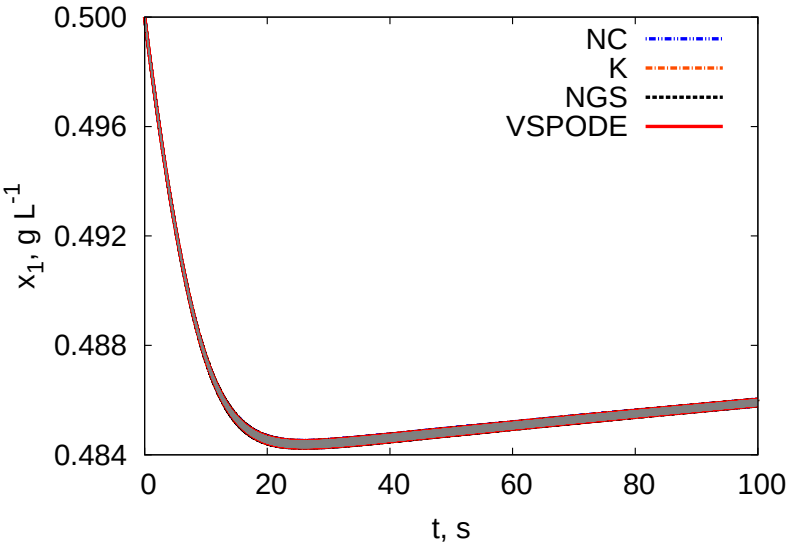


Figure 3.18: State variable x_1 of reactor separator model

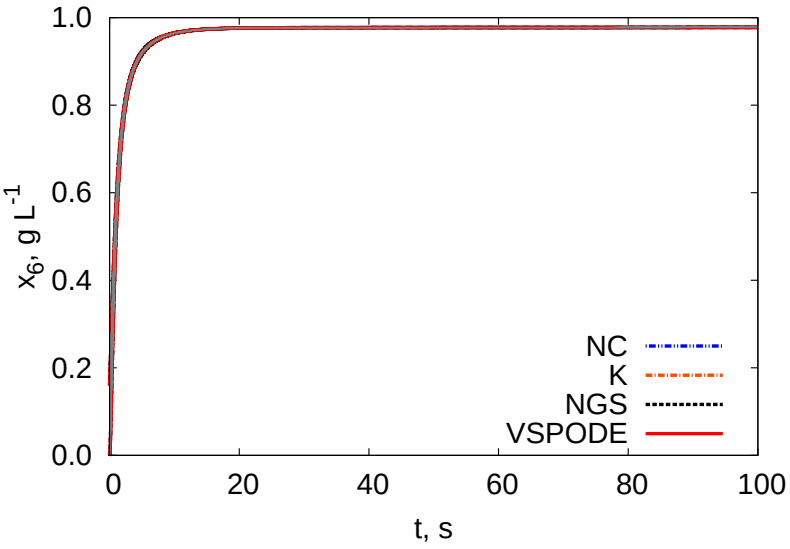


Figure 3.19: State variable x_6 of reactor separator model

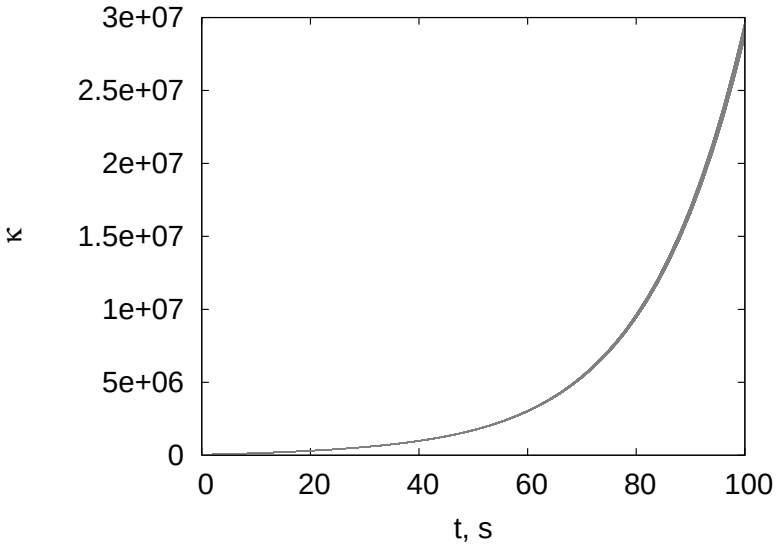


Figure 3.20: Condition number of Glucagon receptor model across the time horizon and the parametric uncertainties

3.5.7 Glucagon receptor model

The mathematical model and the parameters of the glucagon receptor model are presented in Equation (3.32) and Table 3.18, respectively. Its condition number is shown in Figure 3.20. This number quickly increases towards the end making the problem ill-conditioned at this point.

$$
\begin{aligned}
\dot{R}_r &= k_{-1} L R_u - L k_1 R_r - k_s R_r + k_r R_s \\
\dot{R}_s &= k_{sp} L R_p + G_i K_{2s} L R_u + k_s (L R_u + R_r) - k_r R_s \\
\dot{G}_i &= -G_i K_{23} L R_u + G_* \left(k_h + \frac{Ca\, k_{Gdeg,Cal}}{K_{Gdeg,Cal} + G_*} + \frac{PLC_*\, k_{Gdeg,PLC}}{K_{Gdeg,PLC} + G_*} \right) \\
\dot{LR_p} &= -k_{sp} L R_p + k_p \left(1 + \frac{A_0}{1 + B_1 G_*^{-n_1}} \right) \left(\frac{L R_u}{L R_u + B_2} \right) \\
P\dot{L}C_* &= k_{PC} G_* - \frac{PLC_* k_{PC,deg}}{K_{PC,deg} + PLC_*} \\
G_* &= G_0 - G_i \\
R_0 &= R_r + R_s + L R_u + L R_p
\end{aligned}
\tag{3.32}
$$

Table 3.19 shows the results of the four methods. As in the previous case, a single iteration of the methods using contractors took more CPU time than the VSPODE method, so the NC, K and NGS methods were used with the best settings. However, in this case, different

Table 3.18: Initial conditions and system parameters for the Glucagon receptor model problem

Initial conditions and system parameters
$k_1 = 100$
$k_s = 5.2 \times 10^{-3}$
$k_{sp} = k_s$
$K_{2s} = 2 \times 10^{-8}$
$k_r = 4 \times 10^{-3}$
$K_{23} = 1 \times 10^{-7}$
$k_h = 2 \times 10^{-1}$
$k_{Gdeg,Cal} = 1.47 \times 10^3$
$K_{Gdeg,Cal} = 3.54 \times 10^1$
$k_{Gdeg,PLC} = 2.19 \times 10^3$
$K_{Gdeg,PLC} = 5.7$
$k_p = 6.5 \times 10^4$
$A_0 = 3$
$k_{-1} = 10$
$n_1 = 1$
$B_2 = 1 \times 10^6$
$R_0 = 5.5 \times 10^4$
$G_0 = 1 \times 10^5$
$k_{pc} = 6.06 \times 10^{-4}$
$k_{PC,deg} = 2.82 \times 10^{-1}$
$K_{PC,deg} = 2.55 \times 10^{-1}$
Uncertain values
$B_1 = [98.8, 102.2]$

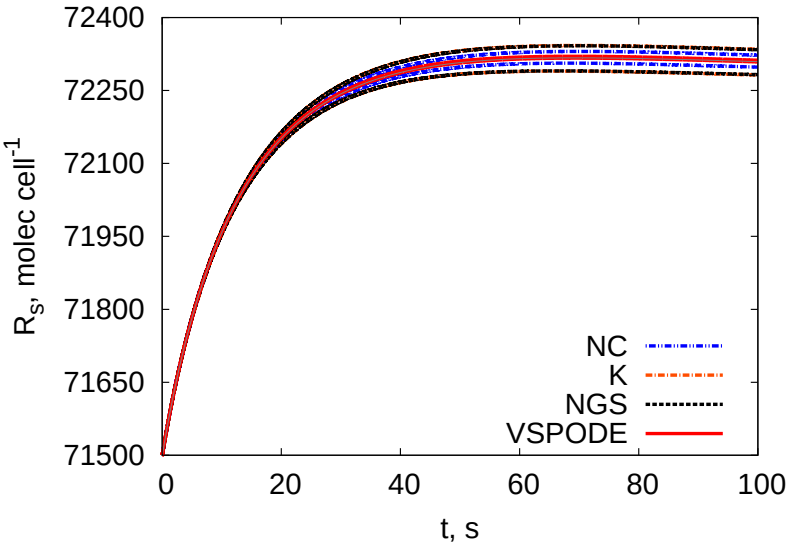


Figure 3.21: State variable R_s of Glucagon receptor model

Table 3.19: Results on the implementation of interval contractors in the Glucagon receptor model

Method	**Contractors**	**CPU(s)**	**Boxes**	**Width at $t_s = 100$ s**
NC	0	40.610	4	$w(R_s(t_s))$=24.5 $w(PLC_*(t_s))$=0.0001216
K	3	42.115	1	$w(R_s(t_s))$=52.5 $w(PLC_*(t_s))$=0.0002621
NGS	3	34.266	1	$w(R_s(t_s))$=51.1 $w(PLC_*(t_s))$=0.0002517
VSPODE	N/A	8.874	1	$w(R_s(t_s))$=3.4539 $w(PLC_*(t_s))$=0.000006

widths were observed. Figures 3.21 and 3.22 show the trajectories of the four methods where the widths can be appreciated visually. Since the NC and the K methods took the longest CPU time, they will be used as the 100% reference. The NGS and VSPODE methods took 83% and 22% of the time taken by the NC and K methods, respectively. In terms of tightness of the bounds, the least tight methods were the ones using contractors (100%). The NC method obtained widths that represent 50% in both states of the widths of the methods using contractors. The VSPODE method yielded widths of $w_{VSPODE}(R_s(t_s))/w_{NGS}(R_s(t_s)) = 6.8\%$ in R_s and $w_{VSPODE}(PLC_*(t_s))/w_{NGS}(PLC_*(t_s)) = 2.4\%$ in PLC_* of the widths of the methods using contractors.

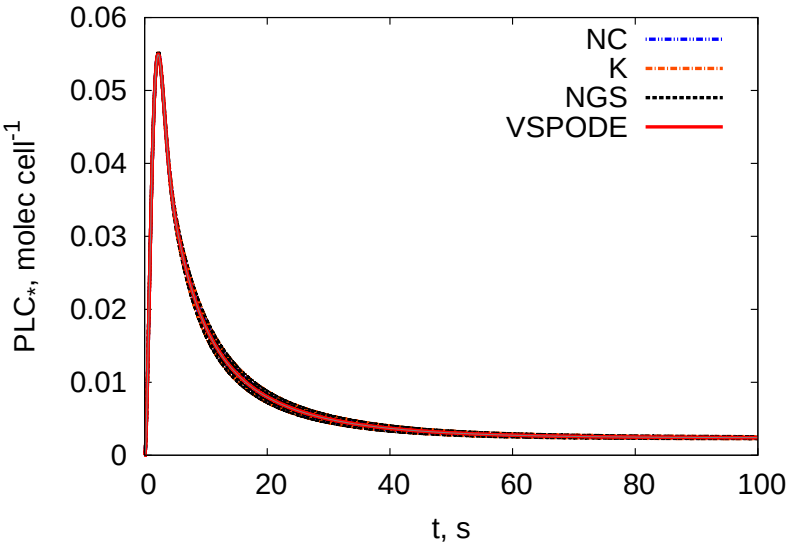


Figure 3.22: State variable PLC_* of Glucagon receptor model

3.6 Conclusions

In this chapter, we have presented two interval contractors (Krawczyk and Newton/Gauss-Seidel) implemented in an implicit procedure in an interval Taylor series method and applied to several case studies from chemical and biochemical processes. The method was compared with and without contractors in seven case studies leading to the conclusion that contractors help reduce the overestimation and are faster than the standalone method in most of the cases.

In terms of the Krawczyk and Newton/Gauss-Seidel contractors. The Newton/Gauss-Seidel contractor performed better than the Krawczyk contractor as it consistently produced tighter widths at almost the same computational time. Furthermore, the glucagon receptor model, provided a better time than the Krawczyk contractor.

The methods with contractors were also compared with the software package VSPODE. Such comparison resulted in that the method performs better in one case study (Exothermic batch reactor). However, in the rest of the case studies, VSPODE showed that the bounds constructed using this Taylor model based solver were tighter.

One of the advantages of the interval Taylor series method is that it provides a quick way to check new methodologies since it is relatively simple to use. For instance, methodologies for other kind of contractors such as constraint propagation can be developed and tested in this method and then take the learnings to more effective but more complicated verified simulation techniques. With the knowledge gained from this chapter, we are ready to test the developed techniques in other kinds of verified simulation techniques, such as Taylor Models and Differential Inequalities.

One of the main disadvantages of of the developed method is that after applying the contractor iterations, the CPU time base method increases almost twice. The reason behind this is that the the contractor iteration is applied at each time step and sometimes it this iteration needed more than once in the same time step. A possible solution for this is to modify the method so that it uses the contractor in a more selective way, only when it is needed instead of every iteration. For instance, not much contracting capability is required at the beginning of the simulation because there has not been enough time for the overestimation to accumulate, which means that the effort can be decreased in the first time steps.

While the method presented is not faster than the method that features Taylor Models (VSPODE), we present a technique that can be applied to methods in which a constraint satisfaction problem can be formulated at every time step. Therefore it can also be applied to other verified solvers that are not necessarily based on interval Taylor series such as other verified simulation methods (for example Taylor Models and convex/concave differential inequalities) and in turn other dynamic simulation methods.

Finally, there is still a challenge in the verified solution method of IVPs for ODEs. The overestimation in these methods is still hindering their ability to bound higher dimensional problems and to manage bigger uncertainties in the parameters in global optimisation. As part of the our future work the investigation of higher dimensional problems is going to be carried out as well as the implementation of constraint propagation contractors and model reformulation techniques to the symbolic expressions in order to avoid the multiple

occurrence of the the same variables.

Chapter 4

Constraint Propagation in Interval Taylor Series

Constraint propagation techniques are widely used in constraint satisfaction problems (CSPs) where the main goal is to eliminate values of variables that do not belong to constraint solutions, i.e., inconsistency. Chapter 4 describes a technique using constraint propagation methods in verified simulation. In particular, the forward-backward contractor is discussed in the context of dynamic simulation through the implementation of an interval Taylor series using an interval forward-backward algorithm (also known as the HC4Revise [5]) to contract the domain of the variables. The proposed method takes advantage of constraints that occur in chemical and biological systems in order to reduce the overestimation generated in the verified simulation. The relevant constraints considered in this chapter are: natural bounds, affine reaction invariants and inequality path constraints from dynamic optimisation. The effectiveness of the method is demonstrated with six numerical case studies.

4.1 Introduction

As previously discussed, addressing the global optimisation problem for dynamic system in a guaranteed way requires of the verified integration of the dynamic constraints. Chapter 3 presented a method for verified integration that provides an enclosure for the dynamic tra-

jectories when there are uncertain initial conditions and system parameters. The method developed aimed at obtaining a better representation of the true solution set by using fixed-point interval contractors to reduce the overestimation and in turn provide better evaluations of the objective function and gradients.

The method presented in Chapter 3 used fixed-point interval contractors (Krawczyk and Newton contractors) at each time step of the integration to reduce the overestimation. The interval contractors method demonstrated to be effective in the problems studied. However, it does not provide a mechanism to further reduce the overestimation when extra information about the system is known such as natural physical bounds, kinetic relations or some other constraints.

An interval contractor that propagates the initial domains when there is extra information about the problem can be implemented in a verified integration algorithm in a similar way as the fixed-point contractors. The forward-backward contractor has been developed in the area of constraint propagation and has been applied to constraint satisfaction and global optimisation problems using either tree or directed acyclic graph (DAG) representations of the mathematical expression.

The forward-backward contractor was presented in Jaulin et al. [27] and in Benhamou et al. [5] with the name HC4Revise. In the latter, the authors also presented an algorithm for performing a forward evaluation and a backward propagation (HC4) in the tree representation of the constraint(s) in which there is no need of decomposing the original constraint(s) into a set of constraints involving only atom operations. The forward-backward contractor has also been considered in the case of monotonic functions. In Chabert and Jaulin [11] an optimal contractor under monotonicity (OCTUM) was described.

Schichl and Neumaier [67] presented a method for performing constraint propagation in DAGs for optimisation. In their method, they show the computation of gradients and slopes for optimisation and they reported their improvement after constraint propagation.

A method that uses a forward-backward propagation algorithm on a partial DAG representation of constraint satisfaction problems based on a branch and prune approach was presented in Vu et al. [82]. Their method can be applied partially because it can represent the active constraints separately. The authors reported that it outperforms by an order of magnitude, a method that uses a forward-backward propagation algorithm in the tree representation of the problem.

In the case of dynamic systems, the work by Jaulin et al. [25] presented a method that combines a forward-backward propagation and a set inversion algorithms [27] for state-estimation in discrete-time systems in which the initial states are given by intervals.

The forward-backward algorithm represents an alternative to further reduce the overestimation when more information about the problem is known. The chapter proceeds as follows. First, the interval Taylor series method and the interval forward-backward contractor are described in Sections 4.3 and 4.4. In Section 4.5 we discuss some of the constraints in dynamic systems on which the contractor can be applied. The implementation of the contractor within the verified simulation is then described in Section 4.6. Afterwards, the effectiveness of the method is demonstrated through the application of the method to pertinent case studies in Section 4.7 and conclusive remarks are given in Section 4.8.

4.2 Problem Formulation

This section considers an interval Taylor series method as in Section 3.3 that will later be used for the implementation of an interval forward-backward contractor. The mathematical form of the problem that this method aims to solve is as follows:

$$\dot{\mathbf{y}}(t) = \mathbf{f}(t, \mathbf{y}(t), \boldsymbol{\theta}),\ \mathbf{y}(t_0, \boldsymbol{\theta}) = \mathbf{y}_0(\boldsymbol{\theta}),\ \mathbf{y}_0(\boldsymbol{\theta}) \in [\mathbf{y}_0],\ \boldsymbol{\theta} \in [\boldsymbol{\theta}] \tag{4.1}$$

where $t \in [t_0, t_f]$, $\boldsymbol{\theta}$ represent the time-invariant parameters with $[\boldsymbol{\theta}] = [\underline{\boldsymbol{\theta}}, \overline{\boldsymbol{\theta}}]$, $\mathbf{y}$ represent the vector of state variables, $\mathbf{y}_0(\boldsymbol{\theta})$ are the initial conditions at time t_0 with $[\mathbf{y_0}] = [\underline{\mathbf{y}_0}, \overline{\mathbf{y}_0}]$.

Here, $\mathbf{f}$ is assumed to be $(k-1)$ times continuously differentiable with respect to the state variables.

The additional considerations are equality or inequality constraints that the state variables have to fulfil during the integration and will be propagated at each time step

$$\begin{aligned} \mathbf{g}_1(\mathbf{y}) &= \mathbf{c} \\ \mathbf{h}_1(\mathbf{y}) &\geq \mathbf{d} \end{aligned} \tag{4.2}$$

where $\mathbf{c}$ and $\mathbf{d}$ are constant vectors.

Also as in the previous chapter, the bounding method used consists of two stages which are shown in the next section.

4.3 Interval Taylor Series

The bounding method used in this section consists of two stages. The first stage is the validation of existence and uniqueness of a solution in which also a suitable a priori enclosure and a time step are obtained. According to this approach h_j and $[\tilde{\mathbf{y}}_j]$ must satisfy the following equation:

$$[\tilde{\mathbf{y}}_j] = [\mathbf{y}_j] + \sum_{i=1}^{k-1} [0, h_j]^i \mathbf{f}^{[i]}([\mathbf{y}_j], [\boldsymbol{\theta}]) + [0, h_j]^k \mathbf{f}^{[k]}([\tilde{\mathbf{y}}_j^0], [\boldsymbol{\theta}]) \subseteq [\tilde{\mathbf{y}}_j^0] \tag{4.3}$$

where k is the order of the Taylor series expansion, $[\mathbf{y}_j]$ is the vector of tight enclosures of the solutions with ranges in $[\tilde{\mathbf{y}}_j^0]$, $[\boldsymbol{\theta}]$ is the vector of system parameters and $\mathbf{f}^{[i]}$ are the Taylor coefficients.

The second stage involves the computation of a tighter enclosure which consists on the use of a high order Taylor series to refine the solution obtained in the first stage. It satisfies

equation (4.4).

$$
\begin{aligned}
[\mathbf{y}_{j+1}] = \overbrace{\hat{\mathbf{y}}_j + \sum_{i=1}^{k-1} h_j^i \mathbf{f}^{[i]}(\hat{\mathbf{y}}_j, \hat{\boldsymbol{\theta}})}^{\mathbf{u}_{j+1}} + \overbrace{\left\{ \mathbf{I} + \sum_{i=1}^{k-1} h_j^i \frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{y}}([\mathbf{y}_j], [\boldsymbol{\theta}]) \right\}}^{[\mathbf{S}_{j+1}^y]} ([\mathbf{y}_j] - \hat{\mathbf{y}}_j) \\
+ \underbrace{\left\{ \sum_{i=0}^{k-1} h_j^i \frac{\partial \mathbf{f}^{[i]}}{\partial \boldsymbol{\theta}}([\mathbf{y}_j], [\boldsymbol{\theta}]) \right\}}_{[\mathbf{S}_{j+1}^\theta]} ([\boldsymbol{\theta}] - \hat{\boldsymbol{\theta}}) + \underbrace{h_j^k \mathbf{f}^{[k]}([\tilde{\mathbf{y}}_j], [\boldsymbol{\theta}])}_{[\mathbf{z}_{j+1}]}
\end{aligned}
\tag{4.4}
$$

where $\mathbf{I}$ is the identity matrix and $\hat{\mathbf{y}}_j = (\underline{\mathbf{y}_j} + \overline{\mathbf{y}_j})/2$. For the sake of simplicity, we rewrite equation (4.4) as

$$
[\mathbf{y}_{j+1}] = \mathbf{u}_{j+1} + [\mathbf{S}_{j+1}^y]([\mathbf{y}_j] - \hat{\mathbf{y}}_j) + [\mathbf{S}_{j+1}^\theta]([\boldsymbol{\theta}] - \hat{\boldsymbol{\theta}}) + [\mathbf{z}_{j+1}] \tag{4.5}
$$

Finally, the interval matrix-vector product $[\mathbf{S}_{j+1}^y]([\mathbf{y}_j] - \tilde{\mathbf{y}}_j)$ in equation (4.5) which is known to be one of the main contributors of the wrapping effect is reduced by rewriting the operations in the following way.

$$
[\mathbf{y}_{j+1}] = \mathbf{u}_{j+1} + ([\mathbf{S}_{j+1}^y]\mathbf{A}_j)[\boldsymbol{\Gamma}_j] + [\mathbf{S}_{j+1}^\theta]([\boldsymbol{\theta}] - \hat{\boldsymbol{\theta}}) + [\mathbf{z}_{j+1}] \tag{4.6}
$$

where,

$$
[\boldsymbol{\Gamma}_{j+1}] = \mathbf{A}_{j+1}^{-1}([\mathbf{z}_{j+1}] - \hat{\mathbf{z}}_{j+1}) + \mathbf{A}_{j+1}^{-1}([\mathbf{S}_{j+1}^y]\mathbf{A}_j)[\boldsymbol{\Gamma}_j] + (\mathbf{A}_{j+1}^{-1}[\mathbf{S}_{j+1}^\theta])([\boldsymbol{\theta}] - \hat{\boldsymbol{\theta}}) \tag{4.7}
$$

Here, $\boldsymbol{\Gamma}_0 = [\mathbf{y}_0] - \hat{\mathbf{y}}_0$, $\mathbf{A}_0 = \mathbf{I}$ and $\mathbf{A}_{j+1}$ is chosen as the orthogonal matrix in the QR factorization of $\hat{\mathbf{S}}_{j+1}^y \mathbf{A}_j$, the parallelepiped enclosure of $[\boldsymbol{\Gamma}_{j+1}]$. This form of rearranging the matrix-vector product was first reported by [36].

4.4 Interval Forward-Backward Contractor

This section describes the second building block of the method: the interval forward-backward contractor. The implementation of the contractor requires the computational representation of the mathematical problem. The tree and directed acyclic graph (DAG)

representations are defined first.

A mathematical expression can be represented by a tree in which every node is either a variable, a constant or an elementary operation. The root node is represented by a p-ary relation symbol and each node but the root node is associated with two intervals, one for forward evaluation and one for backward propagation [5].

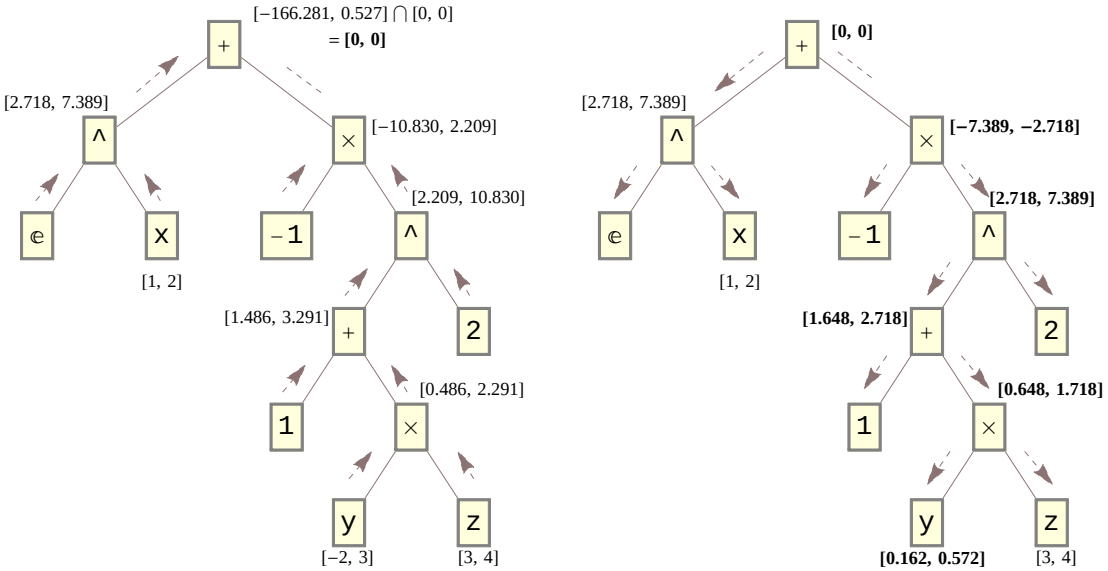
A mathematical expression can also be represented by a DAG. In a DAG, every elementary operation is represented by a node, every constant or variable is a leaf and the edges are directed, each one is associated with a result and an argument node. A DAG contains at least one source and one sink and each node can be associated with two or more intervals.

In this work, a tree representation is used since it is simpler. Directions for using DAGs will be given in the concluding remarks. As the name implies, the contractor is based on constraint propagation in a forward and backward manner. Given relevant constraints for the problem in consideration, a forward evaluation in the tree of the mathematical expression is carried out, updating the bounds in the intermediate nodes and then a second evaluation is done in the opposite direction to update the initial domains of the variables.

The interval forward-backward contractor aims to contract domains of the constraint satisfaction problem (CSP) as in (3.10)

$$(\mathbf{f}(\mathbf{y}) = \mathbf{0},\ \mathbf{y} \in [\mathbf{y}_j])$$

by taking into account one of the $f_i = 0$ constraints in isolation [27]. The contractor proceeds by decomposing every constraint $f_i = 0$ into a sequence of operations involving only basic operations, e.g., $+$, $-$, $\times$, $\div$, *exp*, *log* and *sin*, among others. This gives rise to a list of constraints that involve only one basic operation. These operations also represent the nodes in the expression tree in which it is possible to navigate so as to propagate the constraints by obtaining the natural interval extension.



(a) Forward propagation (b) Backward propagation

Figure 4.1: Forward and backward propagation in expression tree

Consider the the following example.

Example 1.

$$f = e^x - (1 + yz)^2 = 0$$

with domains $[x] = [1, 2]$, $[y] = [-2, 3]$ and $[z] = [3, 4]$. f can be decomposed into a list of constraints that contains only basic operations. The idea is to contract each of these constraints until the contractions become inefficient. The first column of Table 4.1 shows the list of constraints with only basic operations for Example 1; column 2 shows the values of each of the forward constraints (note that the domain in the last constraint has been contracted), and the backward constraints and evaluations are given in columns 3 and 4, respectively. The intervals in bold represent the contracted intervals where it can be appreciated a significant reduction in the domain of the variable y. The forward and backward propagation of the basic constraints can also be appreciated in the expression tree of Example 1 in Figures 4.1a and 4.1b, respectively. The arrows indicate the direction of the propagation and again the intervals in bold represent the reduced domains. The forward and backward propagations are iterated with the updated domain until no further improvement is observed. The forward-backward contractor is also known as HC4Revise.

Table 4.1: Forward and backward evaluation for list of basic operations in example 1

Forward		**Backward**	
$C_1 = e^x$	[2.718, 7.389]	$C_1 = (F - C_5) \cap C_1$	[2.718, 7.389]
$C_2 = yz$	[0.486, 2.291]	$C_5 = (F - C_1) \cap C_5$	**[-7.389, -2.718]**
$C_3 = C_2 + 1$	[1.486, 3.291]	$C_4 = C_5/(-1) \cap C_4$	**[2.718, 7.389]**
$C_4 = C_3^2$	[2.209, 10.830]	$C_3 = C_4^{0.5} \cap C_3$	**[1.648, 2.718]**
$C_5 = -1 \cdot C_4$	[-10.830, -2.209]	$C_2 = C_3 - 1 \cap C_2$	**[0.648, 1.718]**
$F = C1 + C_5$	[-8.112, 5.179]	$Z = C2/Y \cap Z$	[3, 4]
$F = F \cap \{0\}$	[-166.281,0.527]∩[0,0]	$Y = C_2/Z \cap Y$	**[0.162, 0.572]**
	=**[0,0]**	$X = \log C_1 \cap X$	[1, 2]

4.5 Types of Constraints in ODEs

Numerical constraints occur in dynamic systems, for instance mass and energy conservation, chemical kinetics, safety critical applications, among others. This section explores three types of such constraints for constraint propagation applications. The relations that are considered in the next three sections are: natural bounds, affine reaction invariants and inequality path constraints. Of course, there are other types of constraints that can be explored in a similar fashion.

4.5.1 Natural Bounds

Natural bounds appear in dynamic systems as a result of mass and energy conservation balances in some chemical or biological processes. From this consideration, upper bounds can be obtained on a certain concentration or temperature if we have prior knowledge about the process. Natural bounds can also be obtained from physical constraints for example the non-negativity of physical amounts such as concentration, temperature, flow rate, mass and volume, among others, have associated a lower bound of 0.

Example 2. First-order irreversible series reaction [17].

The following reaction

$$A \xrightarrow{k_1} B \xrightarrow{k_2} C$$

is being carried out in a batch reactor. Only the concentrations of A and B were measured and their balance equations are

$$\begin{aligned} \dot{C}_A &= -k_1 C_A \\ \dot{C}_B &= k_1 C_A - k_2 C_B \\ C_A(0) &= 1,\ C_B(0) = 0,\ t \in [0,1] \end{aligned} \tag{4.8}$$

In this example, the concentrations of A and B in the equations are relative to the constant total number of moles. An upper bound on these two states can then be obtained as it is known that the maximum value they can attain is 1. Furthermore, concentrations are physical amounts that cannot take negative values so lower bound for both concentrations is 0. Therefore, upper and lower bounds can be obtained from natural bounds as

$$C_A \in [0,1]$$
$$C_B \in [0,1]$$

4.5.2 Affine Reaction Invariants

In some kinetic models, it is possible to find linear combinations of state variables that are not affected by the chemical reactions taking place in the system, i.e., the affine reaction invariants. The affine reaction invariants can be obtained if the stoichiometry of the system is known and they exist if the stoichiometry matrix is not full row rank. A method to find the reaction invariants and to use them to enclose the trajectories of kinetic models using differential inequalities was described in Scott and Barton [68].

Let us consider a kinetic model of the form

$$\dot{\mathbf{y}}(t,\boldsymbol{\theta}) = \mathbf{D}\mathbf{w}(t,\mathbf{y}(t,\boldsymbol{\theta})) \quad \mathbf{y}(t_0,\boldsymbol{\theta}) = \mathbf{y}_0(\boldsymbol{\theta}) \tag{4.9}$$

where $\mathbf{D}$ is the stoichiometry matrix and $\mathbf{w}$ is the vector of reaction rate functions. In this model, $\mathbf{y}$ represents the state variables, $\boldsymbol{\theta}$ the system parameters and $\mathbf{y}_0$, the initial conditions. The initial conditions and the system parameters take values in the intervals $[\mathbf{y}_0]$ and $[\boldsymbol{\theta}]$, respectively.

The affine reaction invariants can be found by obtaining every vector that lies in the left null space of the stoichiometry matrix, $\mathcal{N}(\mathbf{D}^T)$.

Example 3. The reversible chemical reaction [68]

$$A + B \leftrightharpoons C$$

takes place in an isothermal batch reactor. The kinetic model that describes the change of the concentration with respect to time is

$$\begin{aligned} \dot{y}_A &= -k_1 y_A y_B + k_{-1} y_C \\ \dot{y}_B &= -k_1 y_A y_B + k_{-1} y_C \\ \dot{y}_C &= k_1 y_A y_B - k_{-1} y_C \end{aligned} \tag{4.10}$$

It is of the form (4.9) with

$$\mathbf{D} = \begin{pmatrix} -1 & 1 \\ -1 & 1 \\ 1 & -1 \end{pmatrix}, \quad \mathbf{w}(t, \mathbf{y}(t, \boldsymbol{\theta})) = \begin{pmatrix} k_1 y_A y_B \\ k_{-1} y_C \end{pmatrix} \tag{4.11}$$

From this matrix, it is possible to obtain the left null space or cokernel, i.e, all vectors $\mathbf{x}$ of $\mathbf{S}$ such that $\mathbf{x}^T\mathbf{S} = \mathbf{0}^T$, and from each vector an affine reaction invariant is obtained.

$$\mathcal{N}(\mathbf{D}^T) = \begin{pmatrix} 1 & 1 & 2 \\ 1 & 1 & 0 \end{pmatrix} \tag{4.12}$$

According to the previous matrix, the affine reaction invariants are:

$$\begin{aligned} x_A + x_B + 2x_C &= x_A(0) + x_B(0) + 2x_C(0) \\ x_A - x_B &= x_A(0) - x_B(0) \end{aligned} \tag{4.13}$$

4.5.3 Inequality Path Constraints

In dynamic optimisation problems addressing inequality path constraints is of interest; for example, in safety critical applications where the variable of interest has to meet some specification. These constraints limit the dynamic trajectories in a given time horizon. In dynamic optimisation using control-vector parametrisation, the dynamic system is integrated and a complete search method such as a branch-and-bound framework is used to solve the nonlinear programming (NLP) problem. Normally, it is at the latter stage that the constraint propagation is carried out, for example as in Lin and Stadtherr [32]. There is also the alternative, which is considered here, propagating those constraints from the integration which in turn discards part of the search space and removes the need of an extra algorithm to do the propagation in the optimisation stage.

Example 4. The following second-order exothermic reaction is carried out in an isothermal semibatch reactor.

$$A + B \xrightarrow{k} C$$

The case study was taken from a dynamic optimisation formulation with inequality path constraints [83]. In this problem, there is a stream of B being fed, which is the limiting reactant. The decision variable is the volumetric flow rate θ which contains a concentration $x_{B,in}$ of B. The uncertainty that was considered in the simulation was in the parameter corresponding to the decision variable of the optimisation problem, i.e. θ.

$$\begin{aligned}
\min_{\theta(t)} \phi &= x_A(t_f)V(t_f) - x_A(0)V(0) \\
\dot{x}_A &= -kx_Ax_B - \frac{\theta}{V}x_A \\
\dot{x}_B &= kx_Ax_B + \frac{\theta}{V}(x_{B,in} - x_B) \\
\dot{V} &= \theta \\
\theta &= [0, 0.03] \\
x_A(0) = 2,\ x_B(0) &= 0.5,\ V(0) = 0.7 \\
t &\in [0, 20]
\end{aligned} \tag{4.14}$$

The objective of the problem is to maximise the concentration of C at final time, subject to two safety constraints. The first one accounts for a possible cooling failure due to the feeding of B and the second one sets the feed to zero when the maximum volume is attained.

$$\begin{aligned} T_{fail}(t) &= T + x_B \left(\frac{-\Delta H}{\rho c_p} \right) \leq T_{max} \\ V(t) &\leq V_{max} \end{aligned} \tag{4.15}$$

The same inequality path constraints from the optimisation problem are used as the constraints to propagate in the simulation.

4.6 Verified Simulation with Forward-Backward Contractor

This section describes the use of a constraint propagation forward-backward contractor in a verified ODE solver. The motivation of this approach comes from discrete dynamic systems. Jaulin et al. [26] have proposed an algorithm for performing a forward-backward propagation in discrete dynamic systems.

Lin and Stadtherr [32] have also described a constraint propagation strategy for global optimisation of dynamic systems. They consider a relation $c(x)$ which can be an equality $h(x) = 0$ or inequality $g(x) \leq 0$ constraint. The strategy consists on the computation of the bounds of a Taylor model (Chapter 5) $\mathcal{T}_c$ (obtained via the software package VSPODE) of $c(x)$. Later on, a simplified Taylor polynomial expression is obtained. This expression takes into account the first and second order terms (dominant part) of a Taylor polynomial. According to their strategy, the constraint propagation is carried out in (4.16) when the value $|a_i| \geq \omega$, ω being a pre specified small value that helps to avoid division by 0. When the values of $|a_i| < \omega$ and $|b_i| \geq \omega$, the propagation is carried out in (4.17) and when the values of $|a_i| < \omega$ and $|b_i| < \omega$, no propagation is carried out.

$$[\mathcal{T}_c(\boldsymbol{\theta})] = a_i \left([\theta_i] - \hat{\theta}_i + \frac{b_i}{2a_i} \right)^2 - \frac{b_i^2}{4a_i} + [S_i] \tag{4.16}$$

$$[\mathcal{T}_c(\boldsymbol{\theta})] = a_i([\theta_i] - \hat{x}_i)^2 + b_i([\theta_i] - \hat{\theta}_i) + [S_i] \tag{4.17}$$

In this work, the interval Taylor series (as in Section 4.3) is used as the underlying algorithm of the integration. In order to carry out the constraint propagation, the ODE system is defined in a symbolic manner using the symbolic environment provided by the Ibex library. In the method, each of the basic operations are defined in a symbolic way allowing the construction of the Taylor coefficients by means of automatic differentiation. Once the Taylor coefficients have been computed, it is possible then to obtain a symbolic expression of $\mathbf{y}_{j+1}$ in terms of the state variables and the system parameters with a remainder term that guarantees validity of the enclosure at the current time step. This process is illustrated in algorithm form in Table 4.2.

The Ibex library has several contractors that can be applied to functions defined within its symbolic environment. In our implementation, we take advantage of that and apply a forward-backward and an HC4 contractor to the expression of the enclosure at every time step.

The implementation of the forward-backward or HC4 contractors requires extra information of the problem such as numerical constraints that limit somehow the domain of the variables. As discussed in Section 4.5, there are constraints of interest in dynamic systems such as natural bounds from conservation laws and non-negativity, affine reaction invariants from chemical kinetics and inequality path constraints from dynamic optimisation.

4.6.1 Implementation and Third Party Libraries

The methods explained before were implemented in C++ and third party libraries were used for defining the interval type and for performing automatic differentiation. In particular the library PROFIL/BIAS was used for defining the interval type, FADBAD++ for the automatic differentiation and the Ibex library for the constraint propagation in symbolic functions.

PROFIL/BIAS (Programmer's Runtime Optimized Fast Interval Library/Basic Interval Arithmetic Subroutines) is an open source C++ library (http://www.ti3.tuhh.de/keil/profil/

Table 4.2: Algorithm 2

Initialise: $\mathbf{A}_0 = \mathbf{I}$, $[\mathbf{\Gamma}_0] = [\mathbf{y}_0] - \hat{\mathbf{y}}_0$

`ITSContractorFwdBwd`(`In`: $\mathbf{A}_j$, $[\mathbf{\Gamma}_j]$, h_j, $[\tilde{\mathbf{y}}_j]$, $[\mathbf{y}_j]$, $[\boldsymbol{\theta}]$; `Out`: $\mathbf{A}_{j+1}$, $[\mathbf{\Gamma}_{j+1}]$, $[\mathbf{y}_{j+1}]$)

begin

Symbolic functions

$\tilde{\mathbf{y}}_j^{sym} = \texttt{HOE}([\mathbf{y}_j], [\boldsymbol{\theta}], \mathbf{y}_j^{sym}, \boldsymbol{\theta}^{sym})$

$\mathbf{z}_{j+1}^{sym} = h_j^k \mathbf{f}^{[k]}(\tilde{\mathbf{y}}_j^{sym}, \boldsymbol{\theta}^{sym})$

$\mathbf{S}_{j+1}^{y,sym} = \mathbf{I} + \sum_{i=1}^{k-1} h_j^i \frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{y}}(\mathbf{y}_j^{sym}, \boldsymbol{\theta}^{sym})$

$\mathbf{S}_{j+1}^{\theta,sym} = \sum_{i=0}^{k-1} h_j^i \frac{\partial \mathbf{f}^{[i]}}{\partial \boldsymbol{\theta}}(\mathbf{y}_j^{sym}, \boldsymbol{\theta}^{sym})$

$\mathbf{y}_{j+1}^{sym} = \mathbf{u}_{j+1} + \mathbf{S}_{j+1}^{y,sym} \cdot (\mathbf{y}_j^{sym} - \hat{\mathbf{y}}_j) + \mathbf{S}_{j+1}^{\theta,sym} \cdot (\boldsymbol{\theta}^{sym} - \hat{\boldsymbol{\theta}}) + \mathbf{z}_{j+1}^{sym}$

Introduce constraints

$\mathbf{g}(\mathbf{y}_j^{sym}) \leq 0$ and/or $\mathbf{h}(\mathbf{y}_j^{sym}) = 0$

Implement and evaluate contractors

$[\mathbf{y}_j] \supseteq [\mathbf{y}_j^c] = \texttt{CtcHC4}(\mathbf{g}(\mathbf{y}_j^{sym}) \leq 0)$ and/or $[\mathbf{y}_j] \supseteq [\mathbf{y}_j^c] = \texttt{CtcHC4}(\mathbf{h}(\mathbf{y}_j^{sym}) = 0)$

Algorithm in Table 3.4

$(\mathbf{A}_{j+1}, [\mathbf{\Gamma}_{j+1}], [\mathbf{y}_{j+1}], \mathbf{u}_{j+1}) = \texttt{ITSContractor}(\mathbf{A}_j, [\mathbf{\Gamma}_j], h_j, [\tilde{\mathbf{y}}_j], [\mathbf{y}_j^c], [\boldsymbol{\theta}])$

end

index_e.html) to define an interval type that consists of numbers defined by a lower and an upper bound. The library also defines the interval arithmetic operations and interval intrinsic functions for scalar, vectors and matrices.

PROFIL/BIAS defines the type so that every single operation can be treated as an interval. However, every amount needs to be defined as an interval. If a point valued amount needs to be used, then it is defined as a degenerate interval. For example, 0 is defined as [0,0] and 1 as [1,1] and the vector $\{0, 0\}$ is defined as $\{[0, 0], [0, 0]\}$. This library does not support division by 0 so it throws an undefined behaviour and the program has to stop.

FADBAD++ (Forward Automatic Differentiation, Backward Automatic Differentiation) is an open source C++ library (http://www.imm.dtu.dk/~kajm/FADBAD/) to perform automatic differentiation. Specifically the library allows to compute backward automatic differentiation, forward automatic differentiation and Taylor series expansion. The library can be overloaded with types other than doubles, for example, intervals from PROFIL/BIAS can be used to obtain interval derivatives.

Ibex is an open source C++ library (http://www.ibex-lib.org/) for interval constraint satisfaction problems, global optimisation problems and set characterisation problems among others. The library features many tools such as interval contractors which can be used in symbolic functions with the Ibex syntax. PROFIL/BIAS or other libraries can be used for the defining the interval type.

Figure 4.2 represents the program used to implement a forward-backward constraint propagation in an ITS method with interval contractors NGS and K. In the top of the figure (file `funcivp.cpp`) the mathematical model of the initial value problem is defined, the Taylor coefficients ($\mathbf{f}^{[i]}([\mathbf{y}_j], [\boldsymbol{\theta}])$) and the Jacobian $[\mathbf{S}^{y}_{j+1}]$ are computed next in `tcfunc.cpp` and `jacfunc.cpp`. The information obtained in these programs is enough to compute the first and second stages which are the high order enclosure (`HOE.cpp`) and the tighter enclosure (`ITS.cpp`). In parallel with the previous programs, we construct the symbolic ex-

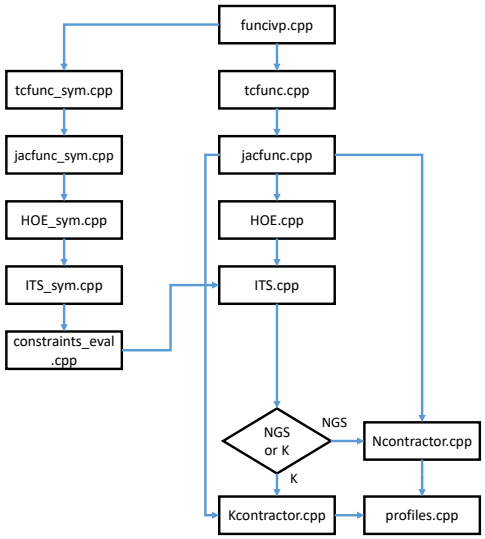


Figure 4.2: Diagram of the C++ programs to implement Forward-Backward constraint propagation in an ITS method with contractors

pressions of the Taylor coefficients `tcfunc_sym.cpp`, the Jacobian of the Taylor coefficients in `jacfunc_sym.cpp` and the steps for the HOE and ITS steps also in symbolic form in `HOE_sym.cpp` and `ITS_sym.cpp`. When the symbolic expression for the ITS is ready, we can implement the constraints and evaluate them in interval arithmetic using the forward-backward contractor. In the Ibex library, when symbolic expressions are evaluated, they stop being symbolic and an interval is vector is obtained. The result of this evaluation is then sent to `ITS.cpp` to be intersected with the current solution of the non-symbolic method. The result of the intersection is normally tighter since the forward-backward contractor is able to discard large regions of overestimation. Later on, the fixed-point contractors are used to refine the solution. The current solution and the information from `jacfunc.cpp` are the inputs to the contractor routines Newton/Gauss-Seidel and Krawczyk, `Ncontractor.cpp` and `Kcontractor.cpp`, respectively. Finally, `profiles.cpp` is in charge of iterating the program and arranging the information of the trajectories.

4.7 Numerical Case Studies

In the present section, numerical experiments to demonstrate the effectiveness of the proposed method are going to be presented. The case studies include one of the constraints

discussed in Section 4.5 and include uncertain amounts in the form of intervals. The implementation for the verified simulation was programmed in C++ and the PROFIL/BIAS [29], FADBAD++ [77] and Ibex [12] third party libraries were used.

In Table 4.3, we revisit the example problem (4.8) in Section 4.5.1 and another case study from [18]. Natural bounds are known for these examples and therefore can be propagated using the proposed method. Table 4.5 presents two case studies [68] in which the affine reaction invariants are used as the constraint to propagate and in turn reduce the overestimation. The case studies in Table 4.7 were taken from a dynamic optimisation formulation with inequality path constraints [83]. The uncertainty that was considered in the simulation was the same as the domain of the decision variable of the optimisation problem, i.e. θ.

For each of the test problems, the figures in this section display the reachable set (shaded grey area) for each case study computed using 4000 Monte Carlo simulations. The trajectories for each method also show the bounds using either the interval forward-backward contractor (FwdBwd) and the interval Newton/Gauss-Seidel contractor (NGS) with a dot dashed orange line, the NGS with a dashed black line and the VSPODE method with a dot dot dashed blue line.

Each test problem subsection also presents the condition number of the problem across the uncertain parameters and time horizon considered. The condition number gives a measure of how sensitive the problem output is when its inputs are perturbed. Problems with small condition numbers are well-conditioned and so small changes in the inputs yield small changes in the outputs. On the other hand, problems with a large condition number are ill-conditioned which means that small changes in the inputs can produce arbitrarily large changes in the outputs. Preliminary, one can expect that problems with large condition to be more challenging when trying to compute upper and lower bounds. The condition number of a differentiable function f can be calculated by:

$$\kappa = \frac{\|J(x)\|}{\|f(x)\|\|x\|} \tag{4.18}$$

Table 4.3: Mathematical models of dynamic simulation case studies with natural bounds

Mathematical model, initial conditions and system parameters	
First Order Irreversible Series Reaction	
$$\dot{C}_A = -k_1 C_A$$ $$\dot{C}_B = k_1 C_A - k_2 C_B$$	
Initial conditions and system parameters	$C_A(0) = 1$, $C_B(0) = 0$
Natural bounds	$C_A(0) = [0, 1]$, $C_B(0) = [0, 1]$
Uncertain values	$k_1 = [4.5, 5.5]$, $k_2 = [0.2, 1.8]$
First Order Reversible Series Reaction	
$$\dot{C}_A = -k_1 C_A + k_{-1} C_B$$ $$\dot{C}_B = k_1 C_A - (k_{-1} + k_2) C_B + k_{-2}(1 - C_A - C_B)$$	
Initial conditions and system parameters	$C_A(0) = 1$, $C_B(0) = 0$, $k_{-1} = 2$, $k_{-2} = 20$
Natural bounds	$C_A(0) = [0, 1]$, $C_B(0) = [0, 1]$
Uncertain values	$k_1 = [2, 6]$, $k_{-1} = [1, 3]$

where J is the Jacobian of f.

4.7.1 Natural Bounds Case Studies

Two problems have been considered to test the algorithm using natural bounds. The test problems are the first order irreversible series reaction and the first order reversible series reaction. the levels of uncertainty are the same as those used in Chapter 3 for both problems. The mathematical form of the problems, the natural bounds and the uncertain parameters are given in Table 4.3.

The graphs with the condition numbers of these two case studies with the same uncertainties have already been presented in Chapter 3 in Figures 3.2 and 3.5. These condition numbers turned out to be not too large so the problems are considered to be well-conditioned.

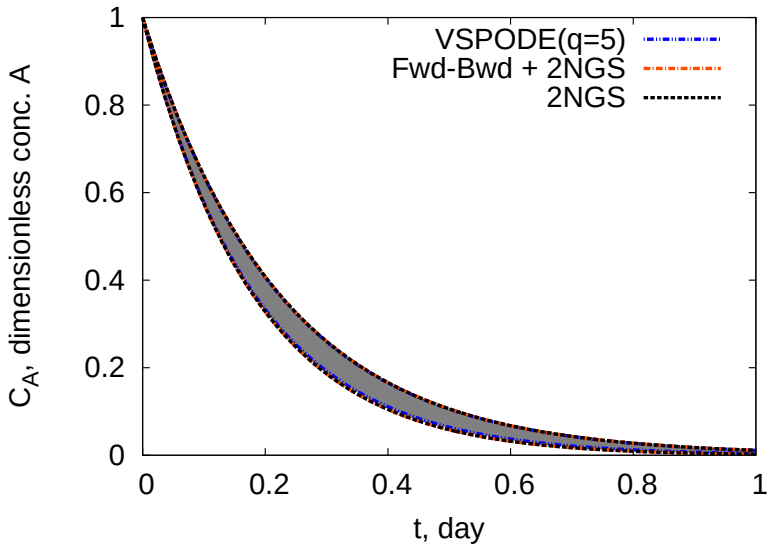


Figure 4.3: State variable C_A of first order irreversible series reaction

Figures 4.3 and 4.4 show the trajectories after using the three methods with natural bounds as additional constraints. Visually, the tightest bounds can be observed in the case of the VSPODE method and the Fwd-Bwd + 2NGS method turned out to be tighter than the newton method alone (2NGS). Table 4.4 reports the widths of the bounds at final time using the three methods. The widest bounds are taken as the the maximum width (in C_A), which in this case belong to the Newton method (100%). Based on this, the widths of the Fwd-Bwd + 2NGS and VSPODE methods correspond to an 84% and 53% in comparison with 2NGS. The trajectories for the first order reversible series reaction using the three methods are presented in Figures 4.5 and 4.6. As in the previous case, the tightest bounds can be observed in the case of the VSPODE method. In this problem the method using the forward backward contractor plus newton seems to be shifted upwards in the C_A state variable due to the natural bounds. Table 4.4 reports the widths of the bounds at final time using the three methods. The fraction of width of the best methods with respect to the method with the largest width have been obtained. The method with widest bounds is 2NGS which corresponds to 100%. The method that follows is Fwd-Bwd+2NGS with 91% in C_A and 66% in C_B. The tightest method is VSPODE with 46% in C_A and 27% in C_B.

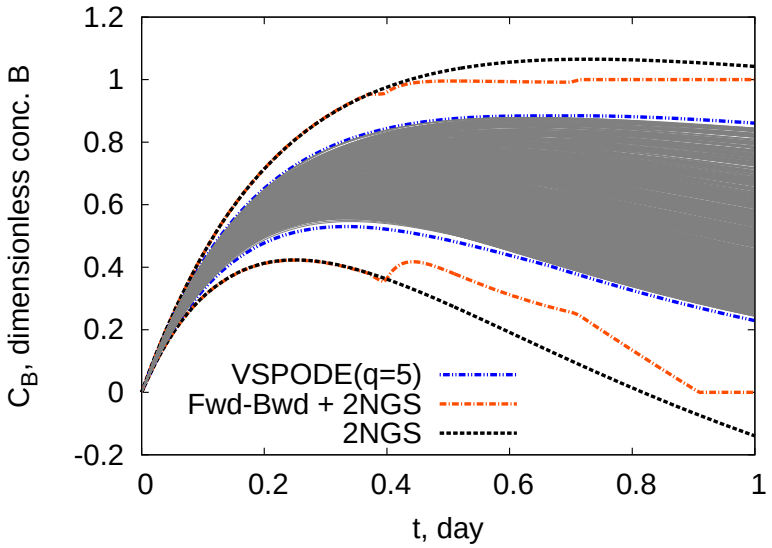


Figure 4.4: State variable C_B of first order irreversible series reaction

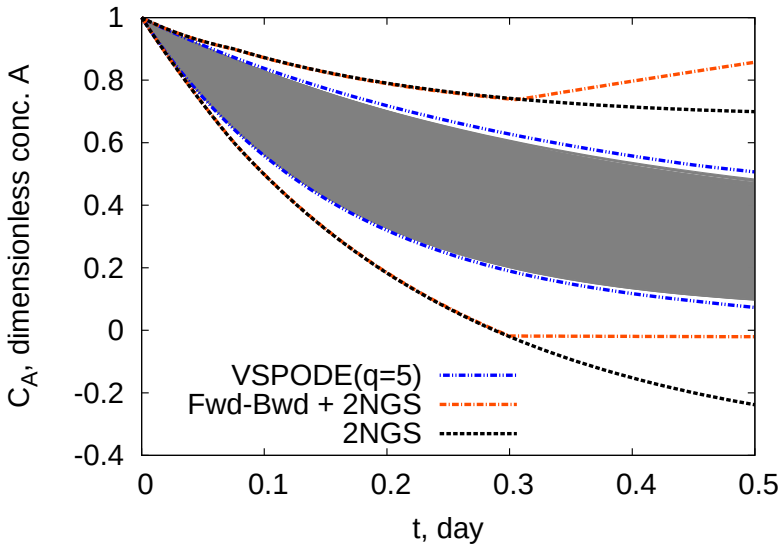


Figure 4.5: State variable C_A of first order reversible series reaction

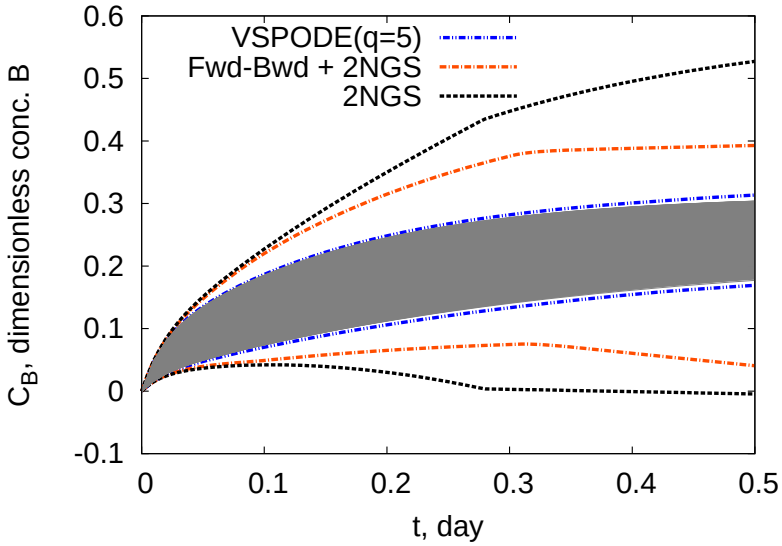


Figure 4.6: State variable C_B of first order reversible series reaction

Table 4.4: Results of the forward-backward constraint propagation method in the natural bounds case studies

Method	**CPU(s)**	**Width at t_f**
First order irreversible series reaction ($t_f = 1$ day)		
2NGS	0.1408	$w(C_A(t_f))$=0.008981 $w(C_B)(t_f)$=1.1864
FwdBwd+2NGS	0.6901	$w(C_A(t_f))$=0.008980 $w(C_B)(t_f)$=1.0000
VSPODE (q=5)	0.0295	$w(C_A(t_f))$=0.007044 $w(C_B)(t_f)$=0.63162
First order reversible series reaction ($t_f = 0.5$ day)		
2NGS	0.1947	$w(C_A(t_f))$=0.9375 $w(C_B)(t_f)$=0.5320
FwdBwd+2NGS	0.7405	$w(C_A(t_f))$=0.8574 $w(C_B)(t_f)$=0.3522
VSPODE (q=5)	0.0143	$w(C_A(t_f))$=0.4334 $w(C_B)(t_f)$=0.14447

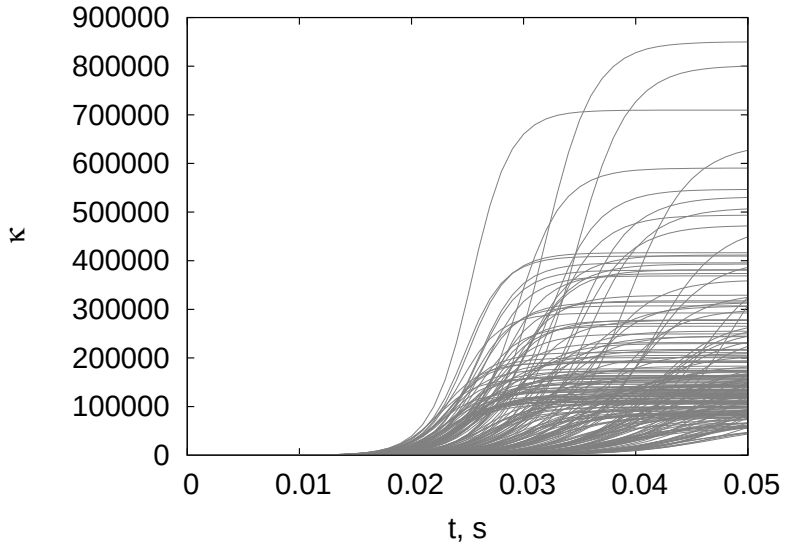


Figure 4.7: Condition number of reversible chemical reaction model across the time horizon and the parametric uncertainties

4.7.2 Affine Reaction Invariants Case Studies

In this section, two problems with affine reaction invariants are addressed. The mathematical models, affine reaction invariants and system parameters of the reversible chemical reaction and chemical kinetics test problems are presented in Table 4.5.

Graphs with the condition numbers of the reversible chemical reaction and the chemical kinetics problems are presented in Figures 4.7 and 4.10. The condition number for the reversible chemical reaction drastically increases from the second half of the integration time. The condition number for the chemical kinetics problem rapidly increases around the final quarter of the simulation time.

Figures 4.8 and 4.9 show the bound obtained with the methods. It can be seen that the Newton method (2NGS) is not able to complete the integration whereas the other two are able to do it. Table 4.6 shows the CPU times and widths of the methods. It can be also seen that the two methods that completed the integration do it in a very different way. VSPODE method bounds the trajectories tightly at the beginning of the integration but seems to be losing control in the end of it. The forward-backward and NGS contractors methods are able to bound the first part of the problem conservatively but finish in a more controlled

Table 4.5: Mathematical models of dynamic simulation case studies with affine reaction invariants

Mathematical model, initial conditions and system parameters	
Reversible Chemical Reaction	$\dot{x}_A = -k_f x_A x_B$ $\dot{x}_B = -k_f x_A x_B + k_r x_C$ $\dot{x}_C = k_f x_A x_B - k_r x_C$
Initial conditions and system parameters	$x_A(0) = 1.5,\ x_B(0) = 0.5,\ x_C(0) = 0$
Affine reaction invariants	$x_A + x_B + 2x_C = x_A(0) + x_B(0) + 2x_C(0)$, $x_A - x_B = x_A(0) - x_B(0)$
Uncertain values	$k_f = [210, 500],\ k_r = [0.001, 0.01]$
Chemical Kinetics	$\dot{x}_A = -k_1 x_A x_B - k_2 x_A x_C$ $\dot{x}_B = -k_1 x_A x_B + k_2 x_A x_C$ $\dot{x}_C = k_1 x_A x_B - k_2 x_A x_C$ $\dot{x}_D = k_2 x_A x_C$
Initial conditions and system parameters	$x_A(0) = 1,\ x_C(0) = 0,\ x_D(0) = 0,\ k_2 = 20$
Affine reaction invariants	$x_A - x_B + 2x_D = x_A(0) - x_B(0) + 2x_D(0)$, $x_B + x_C = x_B(0) + x_C(0)$
Uncertain values	$x_B(0) = [0.95, 1.05],\ k_1 = [50, 500]$

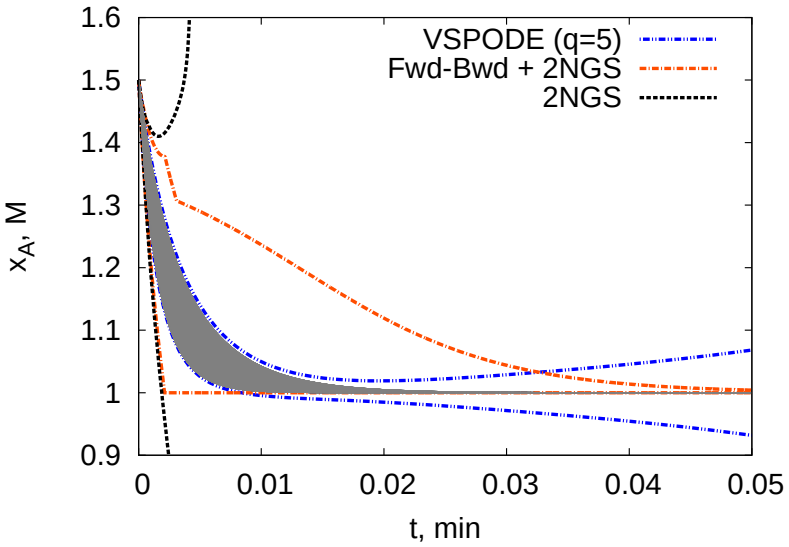


Figure 4.8: State variable x_A of reversible chemical reaction

way with tighter bounds than VSPODE in the final time. Taking the width of the VSPODE as the maximum width (100%), the Fwd-Bwd + 2NGS method has final widths in both variables that represent the 3% of the widths of the VSPODE method.

In the chemical kinetics test problem, only the method using the forward-backward and Newton contractors (Fwd-Bwd + 2NGS) managed to bound the trajectories for the whole time horizon. Figures 4.11 and 4.12 show the bounds of the three methods and the second part of Table 4.6 shows the CPU time and widths for the three methods. Since the 2NGS and the VSPODE methods did not complete the simulation successfully only the time and width for the Fwd-Bwd + 2NGS method are given. The use of the forward-backward contractor in the constraints allowed the construction of bounds that were not possible to obtain with the NGS method alone.

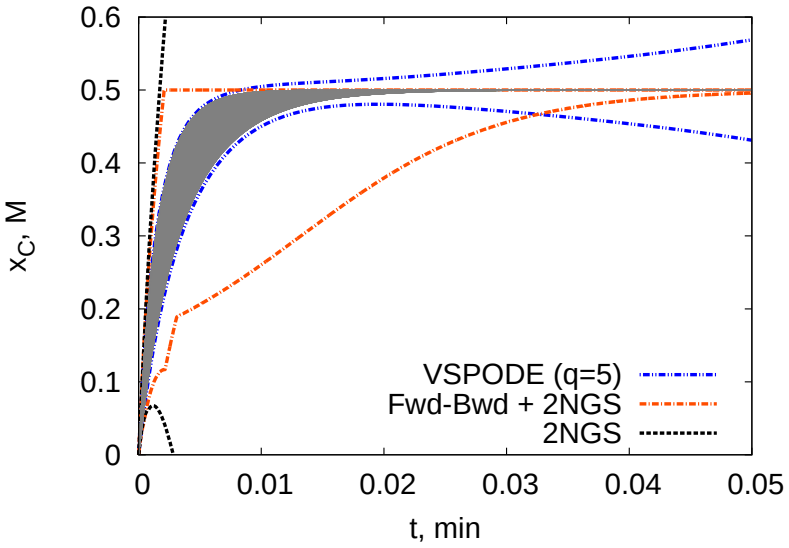


Figure 4.9: State variable x_C of reversible chemical reaction

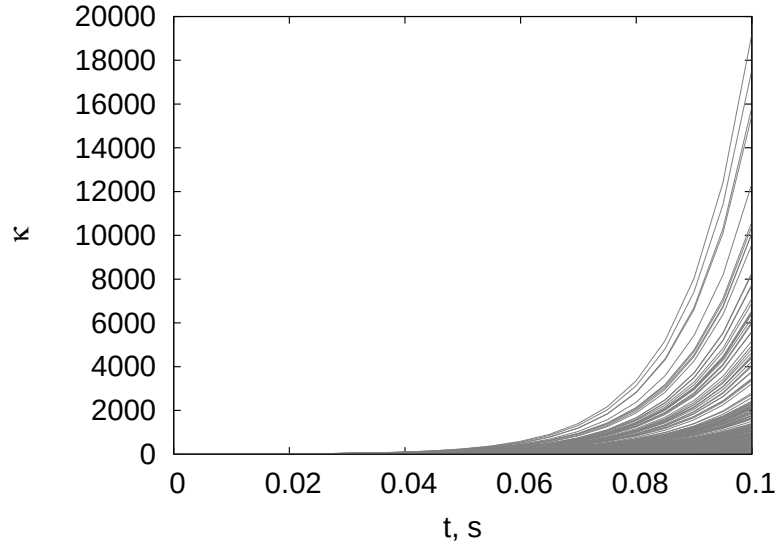


Figure 4.10: Condition number of chemical kinetics model across the time horizon and the parametric uncertainties

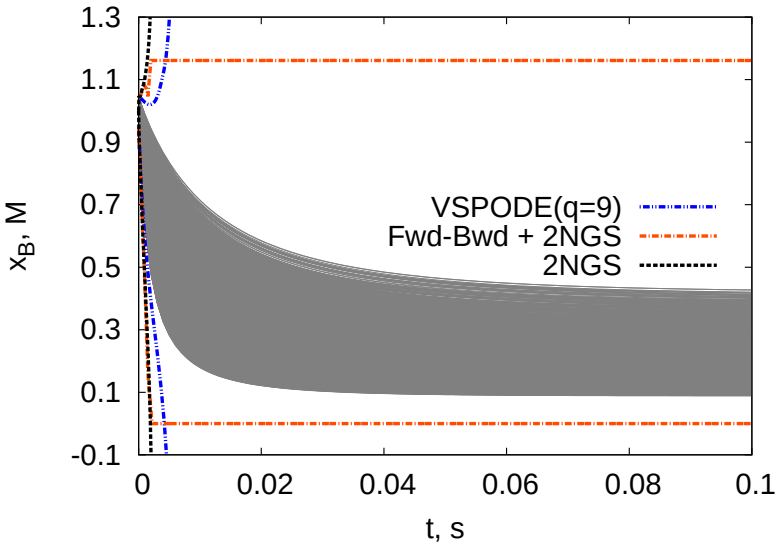


Figure 4.11: State variable x_B of chemical kinetics

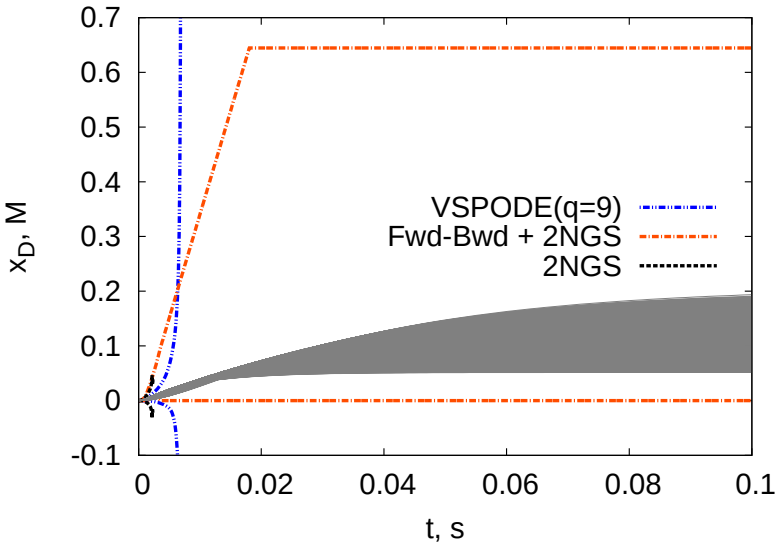


Figure 4.12: State variable x_D of chemical kinetics

Table 4.6: Results of the forward-backward constraint propagation method in the affine reaction invariants case studies

Method	**CPU(s)**	**Width at t_f**
Reversible chemical reaction ($t_f = 0.05$ min)		
2NGS	-	- -
FwdBwd+2NGS	34.3610	$w(x_A(t_f))$=0.004130 $w(x_C(t_f))$=0.004170
VSPODE (q=5)	0.06516	$w(x_A(t_f))$=0.1363 $w(x_C(t_f))$=0.1376
Chemical kinetics ($t_f = 0.1$ s)		
2NGS	-	- -
FwdBwd+2NGS	35.6096	$w(x_B(t_f))$=1.1611 $w(x_D(t_f))$=0.6446
VSPODE (q=9)	-	- -

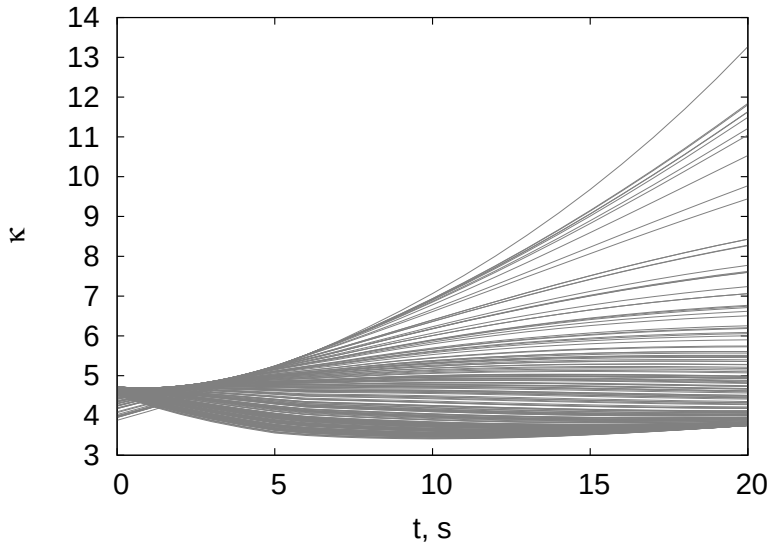


Figure 4.13: Condition number of second-order exothermic reaction in an isothermal semi-batch reactor model across the time horizon and the parametric uncertainties

4.7.3 Inequality Path Constraints Case Studies

Inequality path constraints from optimisation applications are addressed in two problems in this section. The second-order exothermic reaction in an isothermal semibatch reactor and exothermic series reaction in a nonisothermal semibatch reactor. Table 4.7 presents the mathematical model, the inequality path constraints and the uncertain values used. Figures 4.13 and 4.16 show the condition numbers of the two examples. These graphs consider the condition number across the whole time horizon and uncertain values. It can be seen that the two problems are well-conditioned since the condition numbers are not large. In the second-order exothermic reaction in an isothermal semibatch reactor, all the methods tested were able to complete the integration successfully. This can be observed in Figures 4.14 and 4.15. The tightest method turned out to be VSPODE. However, the method with the forward-backward contractor discarded the set of trajectories that are not part of the feasible space in the original optimisation problem. The CPU times and widths of the three methods are reported in Table 4.8. Here, we compare the state x_B in which the inequality path constraint is observed to have the biggest effect. The Newton method (2NGS) has the largest width out of the three methods, so the widths of the other methods are expressed as a percentage of this the maximum width (2NGS). The forward-backward with newton contractor and the VSPODE methods represent the 54% and the 53% of the

Table 4.7: Mathematical models of dynamic simulation case studies with inequality path constraints

Mathematical model, initial conditions and system parameters	
Second-order Exothermic Reaction in an Isothermal Semibatch Reactor	$\dot{x}_A = -kx_Ax_B - \frac{\theta}{V}x_A$ $\dot{x}_B = kx_Ax_B + \frac{\theta}{V}(x_{B,in} - x_B)$ $\dot{V} = \theta$
Initial conditions and system parameters	$x_A(0) = 2,\ x_B(0) = 0.5,\ V(0) = 0.7$
Inequality path constraints	$T + x_B\left(\frac{-\Delta H}{\rho c_p}\right) \leq T_{max},\ V \leq V_{max}$
Uncertain values	$\theta = [0, 0.03]$
Exothermic Series Reaction in a Nonisothermal Semibatch Reactor	$\dot{x}_A = -k_1x_Ax_B - \frac{u}{V}x_A$ $\dot{x}_B = -k_1x_Ax_B + \frac{u}{V}(x_{B,in} - x_B)$ $\dot{x}_C = k_1x_Ax_B - k_2x_C - \frac{u}{V}x_C$ $\dot{V} = u$ $k_i = k_{i,0}e^{\frac{E_i}{RT}} = k_{i,0}e^{\frac{E_i\theta}{R}}$
Initial conditions and system parameters	$x_A(0) = 10,\ x_B(0) = 0,\ x_C(0) = 0,\ V(0) = 1$
Inequality path constraint	$-\Delta H_1k_1x_A(t)x_B(t)V(t) - \Delta H_2k_2x_C(t)V(t) \leq q_{rx,max}$
Uncertain values	$\theta = [3.095, 3.411] \times 10^{-3}$

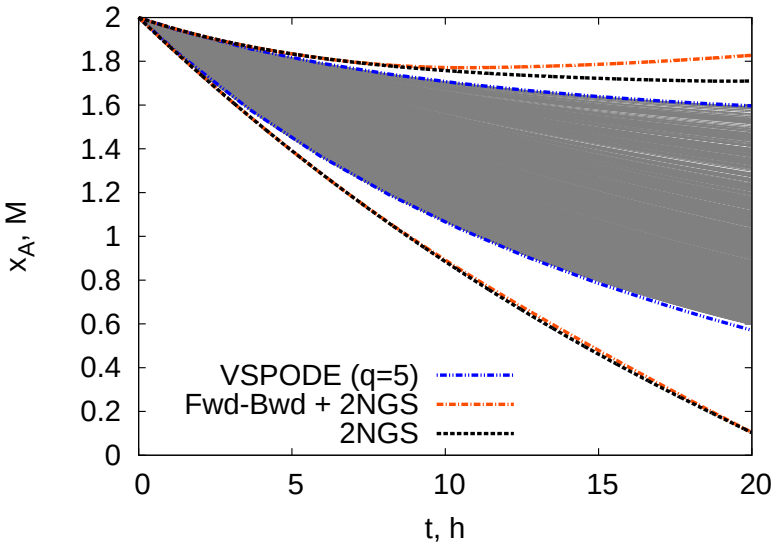


Figure 4.14: State variable x_A of second-order exothermic reaction in an isothermal semi-batch reactor

Newton method width, respectively. In the exothermic series reaction in a nonisothermal semibatch reactor test problem, all the methods were able to complete the simulation up to the final time horizon. The trajectories for the methods are displayed in Figures 4.17 and 4.18. In the figures it can be observed that the tightest method is VSPODE and that the forward-backward with Newton contractor method (Fwd-Bwd + 2NGS) manages to discard a small portion of the trajectories that do not belong to the feasible space in the original optimisation problem. The CPU times and the widths at final time are given in the second part of Table 4.8. Here, the state x_B is the one affected by the inequality path constraint, so we compare the widths at final time in this variable. The method with the widest bounds is the 2NGS method so it represents the 100%. The percentages for the Fwd-Bwd + 2NGS and the VSPODE methods are $w_{Fwd-Bwd+2NGS}(x_B(t_f))/w_{2NGS}(x_B(t_f)) \times 100 = 83\%$ and $w_{VSPODE}(x_B(t_f))/w_{2NGS}(x_B(t_f)) \times 100 = 66\%$, respectively

4.8 Conclusions

We have presented a method to use an interval forward-backward contractor based on constraint propagation through the symbolic computation of the mathematical expressions of the dynamic systems. The method was used in case studies to address constraints such as

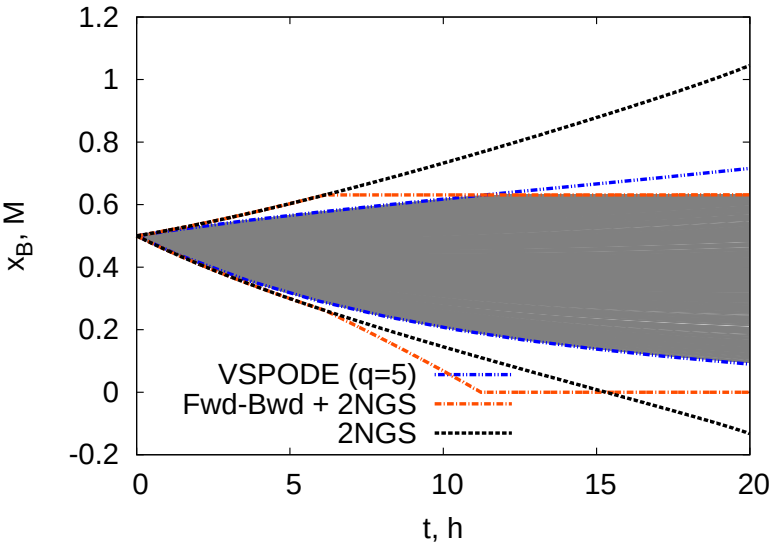


Figure 4.15: State variable x_B of second-order exothermic reaction in an isothermal semi-batch reactor

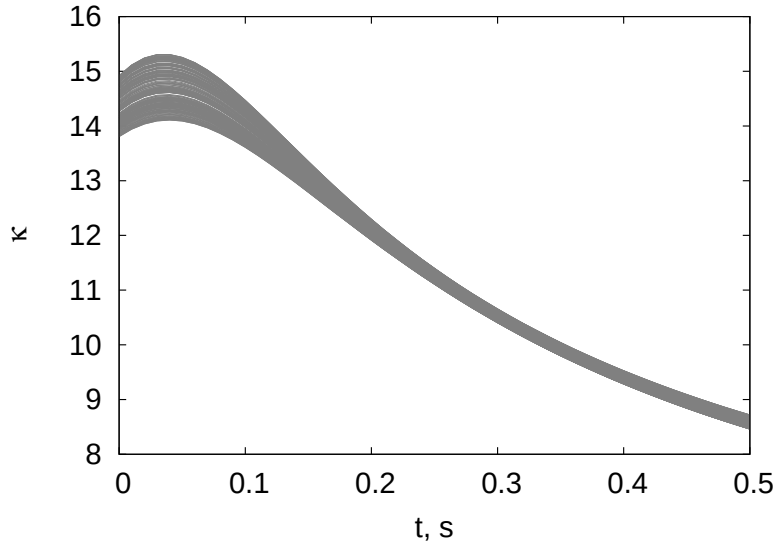


Figure 4.16: Condition number of exothermic series reaction in a nonisothermal semibatch reactor model across the time horizon and the parametric uncertainties

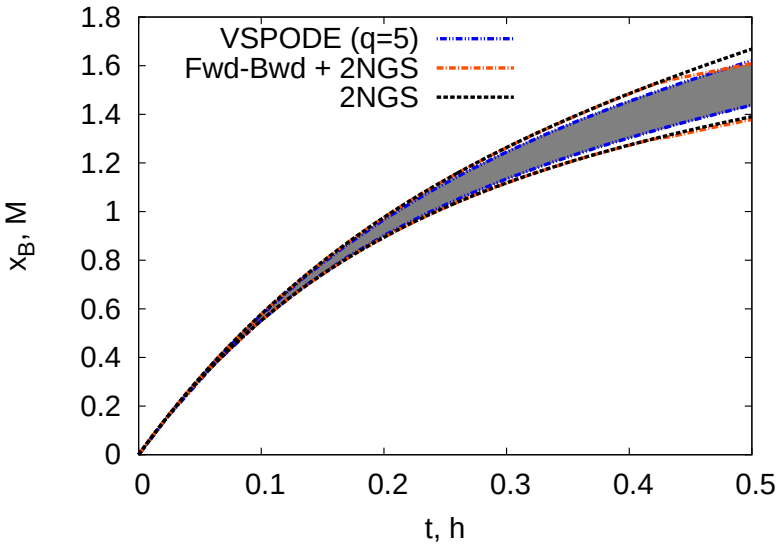


Figure 4.17: State variable x_B of exothermic series reaction in a nonisothermal semibatch reactor

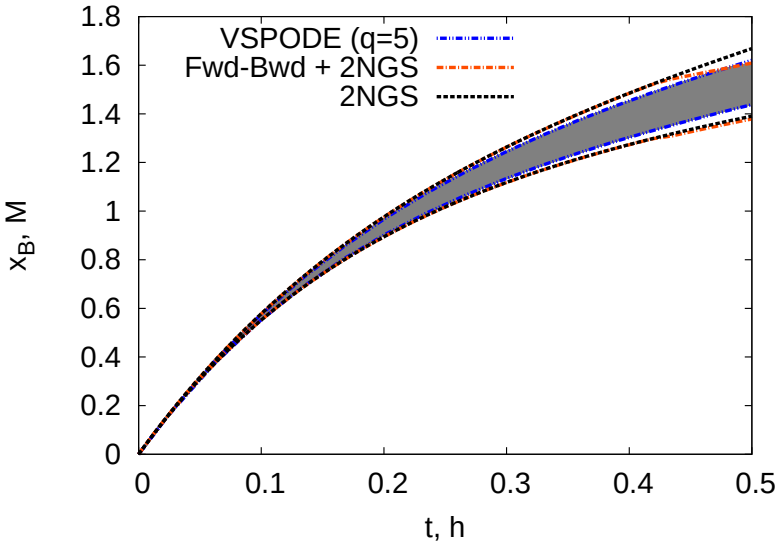


Figure 4.18: State variable x_C of exothermic series reaction in a nonisothermal semibatch reactor

Table 4.8: Results of the forward-backward constraint propagation method in the inequality path constraints case studies

Method	**CPU(s)**	**Width at t_f**
Second-order exothermic reaction ($t_f = 20$ h)		
2NGS	6.6929	$w(x_A(t_f))$=1.6072 $w(x_B(t_f))$=1.1772
FwdBwd+2NGS	16.1229	$w(x_A(t_f))$=1.7224 $w(x_B(t_f))$=0.6312
VSPODE (q=5)	0.03939	$w(x_A(t_f))$=1.02538 $w(x_B(t_f))$=0.6261
Exothermic series reaction ($t_f = 0.5$ h)		
2NGS	1.7906	$w(x_B(t_f))$=0.2779 $w(x_C(t_f))$=0.5544
FwdBwd+2NGS	3.8072	$w(x_B(t_f))$=0.2304 $w(x_C(t_f))$=0.5524
VSPODE (q=5)	0.1725	$w(x_B(t_f))$=0.1826 $w(x_C(t_f))$=0.1087

natural bounds, affine reaction invariants and inequality path constraints.

This kind of approach is favourable when one knows extra information about the problem. In the case of natural bounds, the forward-backward contractor did not produce better bounds than VSPODE. However, the techniques developed in this chapter do reduce the overestimation when used in an interval Taylor series (ITS). The way in which the forward-backward contractor is used in the ITS method can also be exploited and used in other kinds of verified methods such as Taylor models or Differential Inequalities.

The forward-backward contractor takes advantage of known constraints about the problem and provides bounds for the whole time horizons in both of the examples; furthermore, the bounds are improved after using a combination of the forward-backward and Newton/Gauss-Seidel contractors. Combination of contractors have been used in this chapter, but more research is required to determine the right combination of contractors since the order in which they are applied affects the outcome of the problem.

In the area of dynamic optimisation, discarding regions of the decision space at an early stage is of high importance. The method developed here, discards regions of the reachable

set if the inequality path constraint is known. Furthermore, the approach lends itself for application of several constraints, such as affine reaction invariants and natural bounds if they are known.

This chapter presented the condition numbers of all the test problems across the whole time horizon and the uncertain parameters. It was observed that problems with large condition umbers are more challenging to bound because they can become ill-conditioned. More analysis in this regard is needed to fully understand this behaviour.

The biggest disadvantage of the method is that the constraint propagation in the symbolic expressions are slow. An alternative to improve the simulation time is to reduce the number of times the constraint propagations are performed. Currently, the algorithm makes one constraint propagation per time step. However, at the beginning of the simulation, there is not much overestimation, so the constraint propagation can be limited at the beginning and increased later on when there is more need.

Finally, future research directions are the use of directed acyclic graphs (DAGs) instead of tree expressions. Since the approach is able to deal with constraints at the integration stage, the use of the approach in rigorous global optimisation for dynamic systems is promising.

Chapter 5

Interval Contractors in Taylor Models

The present chapter extends the use of interval contractors in interval Taylor series to Taylor models to construct bounds on the trajectories of a given ODE system. The constraint satisfaction problems are used at every time step in a Taylor models approach instead of the interval Taylor series approach. The Newton/Gauss-Seidel contractor is used to reduce the overestimation in a two stage approach. The method showed that interval fixed-point interval contractors are able to reduce the overestimation when there are several levels of uncertainty. Results on the developed algorithm are demonstrated with seven case studies.

5.1 Introduction

Chapter 3 pointed out the relevance of having tight bounds for the dynamic constraints in a dynamic optimisation problem. Apart from the methods based on Tylor series in which the evaluations are done using purely interval arithmetic, there are also polynomial approaches such as Taylor models (or other kinds such as Chebyshev models) that propagate the computations in symbolic form, thus reducing the dependency problem.

Verified methods for solving ODEs using Taylor models have been described in the literature. Berz and Makino [6] described a method for computing the bounds of ODE systems, using remainder differential algebra (RDA) and the Schauder theorem to express the dependence on initial values and time.

Lin and Stadtherr [34] described a method using a two stage Taylor series procedure similar to the interval Taylor series (ITS) approach [42, 47]. In their method, they computed Taylor models of Taylor coefficients using the RDA for the propagation of the operations and for the wrapping effect control, they used the QR decomposition and the parallelepiped procedures. VSPODE, a state of the art software package, incorporates these algorithms.

Sahlodin and Chachuat [65] developed a method similar to Lin and Stadtherr [34] called McCormick-Taylor model. The main difference was in propagating Taylor models with convex/concave remainder using the generalised McCormick relaxation technique [71] instead of interval remainder. The convergence properties of this method were analysed in Bompadre et al. [9] and it seems that the improvement in the bound tightness and the computational expense incurred is problem dependent.

Houska et al. [24] devised a method for verified ODE solution using Taylor models with ellipsoidal remainder bounds. Their algorithm is guaranteed to be stable on infinite time horizons within some uncertainty if the Hausdorff convergence of the extension is quadratic. The authors provided a way to construct such extensions when the set used in the extension is invariant under affine transformation (e.g. polytopes, zonotopes, ellipsoids). In other words, the ellipsoidal remainder term allows the reduction of the overestimation due to the wrapping effect. At every time step in the integration process, when the ellipsoidal enclosure needs to be rotated to wrap the current solution set, it does not generate much overestimation. The reason behind this is that, rotating an ellipsoid preserves collinearity and ratios of distances as opposed to intervals.

Despite these promising advances in the verified solution of ODEs using Taylor models, there is still room for improvement, particularly when introducing more uncertainty and when using higher dimensional models. Additionally, the developments of Chapter 3 have led us to believe that using interval contractors in constraint satisfaction problems is a promising alternative for Taylor models since they are constructed in a similar way as the interval Taylor series method.

In this chapter, Taylor models (Section 5.3) and a verified method for ODEs (Section 5.4) are described. Section 5.5 presents the overestimation reduction technique and a method to implement it. Numerical case studies are presented in Section 5.6 and in Section 5.7, conclusive remarks and proposed future directions are provided.

5.2 Problem Formulation

We assume that the system can be described by an ODE model $\dot{\mathbf{y}}(t) = \mathbf{f}(t, \mathbf{y}(t), \boldsymbol{\theta})$, with $\mathbf{y}$, the vector of state variables and $\boldsymbol{\theta}$, the vector of system parameters. In this chapter, the bold type notation is adopted to indicate vector-valued quantities and square brackets are used for interval-valued quantities unless otherwise specified. The lower and upper endpoints of an interval $[x]$ are specified by $[\underline{x}, \overline{x}]$, the width or diameter of an interval is given by $w([x]) = \overline{x} - \underline{x}$ and the midpoint by $\hat{x} = (\overline{x} - \underline{x})/2$. The mathematical form of the problem that this method aims to solve is as follows:

$$\dot{\mathbf{y}}(t) = \mathbf{f}(t, \mathbf{y}(t), \boldsymbol{\theta}),\ \mathbf{y}(t_0, \boldsymbol{\theta}) = \mathbf{y}_0(\boldsymbol{\theta}),\ \mathbf{y}_0(\boldsymbol{\theta}) \in [\mathbf{y}_0],\ \boldsymbol{\theta} \in [\boldsymbol{\theta}] \tag{5.1}$$

where $t \in [t_0, t_f]$, $\boldsymbol{\theta}$ represent the time-invariant parameters with $[\boldsymbol{\theta}] = [\underline{\boldsymbol{\theta}}, \overline{\boldsymbol{\theta}}]$, $\mathbf{y}$ represent the vector of state variables, $\mathbf{y}_0(\boldsymbol{\theta})$ are the initial conditions at time t_0 with $[\mathbf{y_0}] = [\underline{\mathbf{y}_0}, \overline{\mathbf{y}_0}]$. Here, $\mathbf{f}$ is assumed to be $(k-1)$ times continuously differentiable with respect to the state variables.

A solution of (5.1) from the initial condition $\mathbf{y}_j$ at t_j and parameter values $\boldsymbol{\theta}$ is denoted by $\mathbf{y}(t; t_j, \mathbf{y}_j, \boldsymbol{\theta})$. The solution set of (5.1) with initial conditions $[\mathbf{y}_j]$ at t_j and parameter values $[\boldsymbol{\theta}]$ is given by $\mathbf{y}(t; t_j, [\mathbf{y}_j], [\boldsymbol{\theta}]) = \{\mathbf{y}(t; t_j, \mathbf{y}_j, \boldsymbol{\theta}) \mid \mathbf{y}_j \in [\mathbf{y}_j], \boldsymbol{\theta} \in [\boldsymbol{\theta}]\}$. The objective of this method is to compute an enclosure

$$[\mathbf{y}_j] \supseteq \mathbf{y}(t_j; t_0, [\mathbf{y}_0], [\boldsymbol{\theta}]) = \{\mathbf{y}(t_j; t_0, \mathbf{y}_0, \boldsymbol{\theta}) \mid \mathbf{y}_0 \in [\mathbf{y}_0], \boldsymbol{\theta} \in [\boldsymbol{\theta}]\} \tag{5.2}$$

at $t_j \in [t_0, t_f]$, $j = 1, \ldots N_s$. The bounding method used in this chapter consists of two stages. The first stage is the validation of existence and uniqueness of a solution in which also a suitable a priori enclosure and a time step are obtained. The second stage involves the computation of a tighter enclosure which consists on the use of a high order Taylor series to refine the solution obtained in the first stage.

5.3 Taylor Models

In the method proposed in Chapter 3, the overestimation originated by the dependency problem was not addressed. In Taylor models symbolic and interval computations can be performed since they consist of a polynomial part (symbolic) and an interval remainder bound. Makino and Berz [38, 39] proposed remainder differential algebra (RDA) for bounding function ranges and performing operations on Taylor models using interval arithmetic.

A Taylor model of a function $f(\boldsymbol{\theta})$ is a model comprised by a q^{th} order Taylor polynomial ($\mathcal{P}_f(\boldsymbol{\theta})$) and an interval remainder bound ($[R_f]$). A Taylor model can be represented as

$$f(\boldsymbol{\theta}) \in \mathcal{T}_f(\boldsymbol{\theta}) = \mathcal{P}_f(\boldsymbol{\theta}) + [R_f] = (\mathcal{P}_f(\boldsymbol{\theta}), [R_f]), \forall \ \boldsymbol{\theta} \in \boldsymbol{\Theta} \tag{5.3}$$

RDA can be used to perform arithmetic operations for Taylor models. The polynomial part up to order q of the Taylor model is treated symbolically in the arithmetic operations and the rest of the higher order terms are bounded using interval arithmetic. The contribution of the higher order terms is added to the remainder term.

To illustrate some of the arithmetic operations consider two functions f_1 and f_2 and their Taylor models $\mathcal{T}_{f_1}(\boldsymbol{\theta})$ and $\mathcal{T}_{f_2}(\boldsymbol{\theta})$, respectively.

Binary sum. The Taylor model for the binary sum $f_1 + f_2$ is $\mathcal{T}_{f_1+f_2}(\boldsymbol{\theta}) = \mathcal{T}_{f_1}(\boldsymbol{\theta}) + \mathcal{T}_{f_2}(\boldsymbol{\theta})$

$$\mathcal{T}_{f_1+f_2}(\boldsymbol{\theta}) = (\mathcal{P}_{f_1}(\boldsymbol{\theta}) + \mathcal{P}_{f_2}(\boldsymbol{\theta}), [R_{f_1}] + [R_{f_2}]), \forall \ \boldsymbol{\theta} \in \boldsymbol{\Theta}.$$

Binary product. The Taylor model of the product $f_1 \times f_2$ is $\mathcal{T}_{f_1 \times f_2}(\boldsymbol{\theta}) = \mathcal{T}_{f_1}(\boldsymbol{\theta}) \times \mathcal{T}_{f_2}(\boldsymbol{\theta})$ and is defined as

$$\mathcal{T}_{f_1 \times f_2}(\boldsymbol{\theta}) = \mathcal{P}_{f_1}(\boldsymbol{\theta}) \times \mathcal{P}_{f_2}(\boldsymbol{\theta}) + [\mathcal{P}_{f_1}(\boldsymbol{\theta})] \times [R_{f_2}] + [\mathcal{P}_{f_2}(\boldsymbol{\theta})] \times [R_{f_1}] + [R_{f_1}] \times [R_{f_2}]$$

where $[\mathcal{P}_{f_1}(\boldsymbol{\theta})]$ and $[\mathcal{P}_{f_2}(\boldsymbol{\theta})]$ represent the interval bounds of the polynomials $\mathcal{P}_{f_1}(\boldsymbol{\theta})$ and $\mathcal{P}_{f_1}(\boldsymbol{\theta})$, respectively. In order to preserve the order of the polynomial, the product $\mathcal{P}_{f_1}(\boldsymbol{\theta}) \times \mathcal{P}_{f_2}(\boldsymbol{\theta})$ is separated into two polynomials. That is

$$\mathcal{P}_{f_1}(\boldsymbol{\theta}) \times \mathcal{P}_{f_1}(\boldsymbol{\theta}) = \mathcal{P}_{f_1 \times f_2}(\boldsymbol{\theta}) + \mathcal{P}_e(\boldsymbol{\theta})$$

where $\mathcal{P}_{f_1 \times f_2}(\boldsymbol{\theta})$ includes all terms of order q or less and $\mathcal{P}_e(\boldsymbol{\theta})$ includes the terms of higher order. The contribution of the polynomial $\mathcal{P}_e(\boldsymbol{\theta})$ is bounded and added to the remainder term

$$[R_{f_1 \times f_2}] = [\mathcal{P}_e(\boldsymbol{\theta})] + [\mathcal{P}_{f_1}(\boldsymbol{\theta})] \times [R_{f_2}] + [\mathcal{P}_{f_2}(\boldsymbol{\theta})] \times [R_{f_1}] + [R_{f_1}] \times [R_{f_2}].$$

Finally, the Taylor model of $f_1 \times f_2$ is given by

$$\mathcal{T}_{f_1 \times f_2}(\boldsymbol{\theta}) = (\mathcal{P}_{f_1 \times f_2}(\boldsymbol{\theta}), [R_{f_1 \times f_2}]), \forall \ \boldsymbol{\theta} \in \boldsymbol{\Theta}.$$

Intrinsic functions. The rules for the composition with intrinsic functions have also been described by Makino and Berz [39]. The formulas for the exponential, logarithm, multiplicative inverse, square root, multiplicative inverse of square root, sine, cosine, hyperbolic sine, hyperbolic cosine, arcsine, arccosine and arctangent functions are presented in Appendix 1.

At some point in the computations, either at intermediate stages or at the end, we need to know the range of the Taylor models, i.e., we need to bound them. The easiest way to bound a polynomial is to directly evaluate it using interval arithmetic, however, this only results in very conservative bounds. In this work, the compromise approach described by Lin and Stadtherr [34] is used. In the approach only the first-order and diagonal second-order terms

are bounded exactly and the rest of the terms are bounded directly using interval analysis.

$$[\mathcal{P}_f(\boldsymbol{\theta})] = \sum_{i=1}^{m} (a_i([\boldsymbol{\theta}_i] - \boldsymbol{\theta}_{i0}) + b_i([\boldsymbol{\theta}_i] - \boldsymbol{\theta}_{i0})) + [Q] \tag{5.4}$$

where $[Q]$ contains the rest of the terms of the polynomial $\mathcal{P}_f(\boldsymbol{\theta})$. In order to reduce the dependency problem, (5.4) can be reformulated to have no variable repetitions as

$$[\mathcal{P}_f(\boldsymbol{\theta})] = \sum_{i=1}^{m} \left(a_i \left([\boldsymbol{\theta}_i] - \boldsymbol{\theta}_{i0} + \frac{b_i}{2a_i} \right)^2 - \frac{b_i^2}{4a_i} \right) + [Q]. \tag{5.5}$$

Evaluating (5.5) results in tighter bounds than (5.4).

A useful concept in the following sections is the Taylor model with centred remainder term; this differently reformulated Taylor model can be obtained by modifying the constant part of the polynomial and the remainder term, so that the midpoint of the remainder equals zero ($\hat{R}_f^c = 0$).

$$\begin{aligned} \mathcal{T}_f(\boldsymbol{\theta}) = \hat{\mathcal{T}}_f(\boldsymbol{\theta}) &= (\hat{\mathcal{P}}_f(\boldsymbol{\theta}), [R_f^c]) \\ &= (\mathcal{P}_f(\boldsymbol{\theta}) + \hat{R}_f, [R_f] - \hat{R}_f) \end{aligned} \tag{5.6}$$

5.4 Verified Integration of ODEs. Taylor Models

Just as in the verified method for ODEs using interval Taylor series, a similar method can be described using Taylor models. Lin and Stadtherr [34] and Sahlodin and Chachuat [65] have described a method using a two stage approach similar to the one in Chapter 3 in which stages for validation and tightening are used. The next two sections describe the validation (Section 5.4.1) and tightening (Section 5.4.2) stages.

5.4.1 First Stage. Validation of Existence and Uniqueness

The validation of existence and uniqueness is carried out in the same way as in Chapter 3. An appropriate a priori enclosure ($[\tilde{\mathbf{y}}_j]$) and a time step (h_j) are obtained using the HOE method [48] as in (3.3).

5.4.2 Second Stage. Tightening of the Solution

As in Chapter 3, the second stage involves the computation of a tight enclosure $[\mathbf{y}_{j+1}] \supseteq \mathbf{y}(t_{j+1}; t_0, [\mathbf{y}_0], [\boldsymbol{\theta}])$ given state bounds bounds $[\mathbf{y}_j] = [\mathcal{T}_{\mathbf{y}_j}]$ at t_j. As a starting point, the Taylor model for the initial conditions can be written as $\mathcal{T}_{\mathbf{y}_{0i}} = (\hat{y}_{0i} + (y_{0i} - \hat{y}_{0i}), [0, 0])$.

When a traditional interval Taylor series (ITS) method is used, the Taylor coefficients need to be computed as well. Remainder differential algebra (RDA) as described by Makino and Berz [39] can be used to propagate the automatic differentiation operations in order to get Taylor models of $\mathbf{f}^{[i]}$.

The computation of such bounds is carried out in two different manners depending on which one provides the tightest bounds. In this chapter, the methods used are similar to the two methods described in Sahlodin and Chachuat [65] and Lin and Stadtherr [34].

The following method is similar to the one described by Sahlodin and Chachuat [65]:

$$\begin{aligned}\mathcal{T}_{\mathbf{y}_{j+1}}(\boldsymbol{\theta}) = &\overbrace{\hat{\mathcal{P}}_{\mathbf{y}_j}(\boldsymbol{\theta}) + \sum_{i=1}^{k-1} h_j^i \mathcal{T}_{\mathbf{f}^{[i]}}(\hat{\mathcal{P}}_{\mathbf{y}_j}(\boldsymbol{\theta}), \boldsymbol{\theta})}^{\mathcal{T}_{\mathbf{u}_{j+1}}(\boldsymbol{\theta})} \\ &+ \underbrace{\left\{\mathbf{I} + \sum_{i=1}^{k-1} h_j^i \mathcal{T}_{\frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{y}}}(\mathcal{T}_{\mathbf{y}_j}(\boldsymbol{\theta}), \boldsymbol{\theta})\right\}}_{\mathcal{T}_{\mathbf{S}^y_{j+1}}(\boldsymbol{\theta})} [R^c_{\mathbf{y}_j}] + \underbrace{h_j^k \mathbf{f}^{[k]}([\tilde{\mathbf{y}}_j], [\boldsymbol{\theta}])}_{[\mathbf{Z}_{j+1}]}\end{aligned} \tag{5.7}$$

where $\mathcal{T}_{\mathbf{f}^{[i]}}$ represent the Taylor models of the Taylor coefficients and $\mathcal{T}_{\frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{y}}}$ the Taylor models of the Jacobian of $\mathbf{f}^{[i]}$. In the original method by Sahlodin and Chachuat [65], the term $[\mathbf{Z}_{j+1}]$ is computed using Taylor model whereas in this method, it is computed only interval analysis. According to the authors, this change was found to have a small influence in the solution.

For simplicity, (5.7) is rewritten as

$$\mathcal{T}_{\mathbf{y}_{j+1}}(\boldsymbol{\theta}) = \mathcal{T}_{\mathbf{u}_{j+1}}(\boldsymbol{\theta}) + \mathcal{T}_{\mathbf{S}^y_{j+1}}(\boldsymbol{\theta})[R^c_{\mathbf{y}_j}] + [\mathbf{Z}_{j+1}]. \tag{5.8}$$

To avoid the direct evaluation of $\mathcal{T}_{\mathbf{S}^y_{j+1}}(\boldsymbol{\theta})[R^c_{\mathbf{y}_j}]$ and in turn avoid the wrapping effect, we rewrite (5.8) as

$$\mathcal{T}_{\mathbf{y}_{j+1}}(\boldsymbol{\theta}) = \mathcal{T}_{\mathbf{u}_{j+1}}(\boldsymbol{\theta}) + \left\{\mathcal{T}_{\mathbf{S}^y_{j+1}}(\boldsymbol{\theta})\mathbf{A}_j\right\}\mathcal{T}_{\boldsymbol{\Gamma}_j}(\boldsymbol{\theta}) + [\mathbf{Z}_{j+1}] \tag{5.9}$$

where

$$\begin{aligned}\mathcal{T}_{\boldsymbol{\Gamma}_{j+1}}(\boldsymbol{\theta}) = &\left\{\mathbf{A}^{-1}_{j+1}(\mathcal{T}_{\mathbf{S}^y_{j+1}}(\boldsymbol{\theta})\mathbf{A}_j)\right\}\mathcal{T}_{\boldsymbol{\Gamma}_j}(\boldsymbol{\theta}) \\ &+ \mathbf{A}^{-1}_{j+1}\left\{[\mathbf{Z}_{j+1}] + \mathcal{T}_{\mathbf{u}_{j+1}}(\boldsymbol{\theta}) - \hat{\mathcal{T}}_{\mathbf{y}_{j+1}}(\boldsymbol{\theta})\right\}\end{aligned} \tag{5.10}$$

Tighter enclosure according to Lin and Stadtherr [34]:

$$\mathcal{T}_{\mathbf{v}_{j+1}}(\boldsymbol{\theta}) = \hat{\mathcal{P}}_{\mathbf{y}_j}(\boldsymbol{\theta}) + \sum_{i=1}^{k-1} h^i_j \mathcal{T}_{\mathbf{f}^{[i]}}(\hat{\mathcal{P}}_{\mathbf{y}_j}(\boldsymbol{\theta}), \boldsymbol{\theta}) + [\mathbf{Z}_{j+1}] \tag{5.11}$$

$$\mathcal{T}_{\mathbf{y}_{j+1}}(\boldsymbol{\theta}) = \hat{\mathcal{P}}_{\mathbf{v}_{j+1}}(\boldsymbol{\theta}) + \mathbf{A}_{j+1}\mathcal{T}_{\boldsymbol{\Gamma}_{j+1}}(\boldsymbol{\theta}) \tag{5.12}$$

where

$$\mathcal{T}_{\boldsymbol{\Gamma}_{j+1}}(\boldsymbol{\theta}) = \left\{\mathbf{A}^{-1}_{j+1}(\mathcal{T}_{\mathbf{S}^y_{j+1}}(\boldsymbol{\theta})\mathbf{A}_j)\right\}\mathcal{T}_{\boldsymbol{\Gamma}_j}(\boldsymbol{\theta}) + \mathbf{A}^{-1}_{j+1}[R^c_{\mathbf{v}_{j+1}}] \tag{5.13}$$

5.5 Reduction of the Overestimation via Interval Contractors

5.5.1 Overestimation in Verified Simulation

As in verified simulation using interval Taylor series, overestimation exists when using Taylor models as well. The use of symbolic computations to propagate the polynomial part of the Taylor model contributes to reduce the dependency problem as a result of a decreased number of interval computations in the intermediate stages. However, since the bounding of higher order terms is not done exactly, the dependency problem is still present.

On the other hand, the wrapping effect problem is not further addressed by Taylor models with remainder interval remainder term since the techniques to reduce this problem based on coordinate transformation are the same as those discussed in Chapter 3 (QR decomposition, parallelepiped enclosure).

5.5.2 Fixed-point Interval Contractors

The technique developed in Section 3.4.2 of Chapter 3 proved that when constraint satisfaction techniques (such as the Krawczyk and Newton/Gauss-Seidel contractors) are applied at every time step in the integration, tighter bounds are obtained because the contractor discards part of the region that does not satisfy the constraint and so overestimation reduction is achieved.

The Newton/Gauss-Seidel contractor (Section 3.4.2) will be used in the development of a new method for verified simulation. The method will use the contractor at every time step as in the interval Taylor series method in order to see if reduction of the overestimation is obtained after a number of iterations.

5.5.3 Implementation of Interval Contractors in Taylor Models

In order to implement the interval contractor in the verified method, we formulate a constraint satisfaction problem of the form $f(x) = 0$. In this section, the constraint constructed at every time step is $\mathbf{g}_{j+1}$ and is obtained from the Taylor model of the implicit form of $\mathcal{T}_{\mathbf{y}_{j+1}}(\boldsymbol{\theta}) = f(\mathcal{T}_{\mathbf{y}_j}(\boldsymbol{\theta}))$ as

$$\mathcal{T}_{\mathbf{g}_{j+1}}(\mathcal{T}_{\mathbf{y}_j}(\boldsymbol{\theta}), \boldsymbol{\theta}) = f(\mathcal{T}_{\mathbf{y}_j}(\boldsymbol{\theta})) - \mathcal{T}_{\mathbf{y}_{j+1}}(\boldsymbol{\theta})$$

which can also be rewritten in expanded form as

$$\begin{aligned} \mathcal{T}_{\mathbf{g}_{j+1}}(\mathcal{T}_{\mathbf{y}_j}(\boldsymbol{\theta}), \boldsymbol{\theta}) = \mathcal{T}_{\mathbf{u}_{j+1}}(\boldsymbol{\theta}) + \left\{ \mathcal{T}_{\mathbf{S}^y_{j+1}}(\mathcal{T}_{\mathbf{y}_j}(\boldsymbol{\theta}), \boldsymbol{\theta}) \mathbf{A}_j \right\} \mathcal{T}_{\mathbf{\Gamma}_j}(\mathcal{T}_{\mathbf{y}_j}(\boldsymbol{\theta}), \boldsymbol{\theta}) \\ + [\mathbf{Z}_{j+1}] - \hat{\mathcal{T}}_{\mathbf{y}_{j+1}}(\boldsymbol{\theta}). \end{aligned} \tag{5.14}$$

The formulation of the constraint satisfaction problem requires $\mathcal{T}_{\mathbf{g}_{j+1}}(\mathcal{T}_{\mathbf{y}_j}(\boldsymbol{\theta}), \boldsymbol{\theta})$ to be equal to zero and hence an evaluation at $\mathcal{P}_{\mathbf{y}_j}(\boldsymbol{\theta})$ and $\hat{\mathbf{y}}_j$ is performed in the case of the Taylor model terms and interval term, respectively. This ensures $\mathcal{T}_{\mathbf{g}_{j+1}}(\mathcal{T}_{\mathbf{y}_j}(\boldsymbol{\theta}), \boldsymbol{\theta}) = \mathbf{0}$ when there

is no uncertainty.

$$\begin{aligned}\mathcal{T}_{\mathbf{g}_{j+1}}(\mathcal{P}_{\mathbf{y}_j}(\boldsymbol{\theta}),\boldsymbol{\theta}) = \mathcal{T}_{\mathbf{u}_{j+1}}(\boldsymbol{\theta}) + \left\{\mathcal{T}_{\mathbf{S}^y_{j+1}}(\mathcal{P}_{\mathbf{y}_j}(\boldsymbol{\theta}),\boldsymbol{\theta})\mathbf{A}_j\right\}\mathcal{T}_{\boldsymbol{\Gamma}_j}(\mathcal{P}_{\mathbf{y}_j}(\boldsymbol{\theta}),\boldsymbol{\theta}) \\ + [\mathbf{Z}_{j+1}](\hat{\mathbf{y}}_j,[\boldsymbol{\theta}]) - \hat{\mathcal{P}}_{\mathbf{y}_{j+1}}(\boldsymbol{\theta}).\end{aligned} \tag{5.15}$$

Rearranging (5.15)

$$\begin{aligned}\mathcal{T}_{\mathbf{g}_{j+1}}(\mathcal{P}_{\mathbf{y}_j}(\boldsymbol{\theta}),\boldsymbol{\theta}) = \left\{\mathcal{T}_{\mathbf{S}^y_{j+1}}(\mathcal{P}_{\mathbf{y}_j}(\boldsymbol{\theta}),\boldsymbol{\theta})\mathbf{A}_j\right\}\mathcal{T}_{\boldsymbol{\Gamma}_j}(\mathcal{P}_{\mathbf{y}_j}(\boldsymbol{\theta}),\boldsymbol{\theta}) \\ + [\mathbf{Z}_{j+1}](\hat{\mathbf{y}}_j,[\boldsymbol{\theta}]) + \mathcal{T}_{\mathbf{u}_{j+1}}(\boldsymbol{\theta}) - \hat{\mathcal{P}}_{\mathbf{y}_{j+1}}(\boldsymbol{\theta})\end{aligned} \tag{5.16}$$

and recalling (5.10) from Section 5.4.2

$$\begin{aligned}\mathcal{T}_{\boldsymbol{\Gamma}_{j+1}}(\mathcal{T}_{\mathbf{y}_j}(\boldsymbol{\theta}),\boldsymbol{\theta}) = \left\{\mathbf{A}^{-1}_{j+1}\left(\mathcal{T}_{\mathbf{S}^y_{j+1}}(\mathcal{T}_{\mathbf{y}_j}(\boldsymbol{\theta}),\boldsymbol{\theta})\mathbf{A}_j\right)\right\}\mathcal{T}_{\boldsymbol{\Gamma}_j}(\mathcal{T}_{\mathbf{y}_j}(\boldsymbol{\theta}),\boldsymbol{\theta}) \\ + \mathbf{A}^{-1}_{j+1}\left\{[\mathbf{Z}_{j+1}] + \mathcal{T}_{\mathbf{u}_{j+1}}(\boldsymbol{\theta}) - \hat{\mathcal{T}}_{\mathbf{y}_{j+1}}(\boldsymbol{\theta})\right\}\end{aligned}$$

When (5.10) is evaluated at $\mathcal{P}_{\mathbf{y}_j}(\boldsymbol{\theta})$ and $\hat{\mathbf{y}}_j$, it is very similar to (5.16) and in fact after multiplying it by $\mathbf{A}_{j+1}$, it leads to the same expression (Equation (5.18)).

$$\begin{aligned}\mathcal{T}_{\boldsymbol{\Gamma}_{j+1}}(\mathcal{P}_{\mathbf{y}_j}(\boldsymbol{\theta}),\boldsymbol{\theta}) = \left\{\mathbf{A}^{-1}_{j+1}\left(\mathcal{T}_{\mathbf{S}^y_{j+1}}(\mathcal{P}_{\mathbf{y}_j}(\boldsymbol{\theta}),\boldsymbol{\theta})\mathbf{A}_j\right)\right\}\mathcal{T}_{\boldsymbol{\Gamma}_j}(\mathcal{P}_{\mathbf{y}_j}(\boldsymbol{\theta}),\boldsymbol{\theta}) \\ + \mathbf{A}^{-1}_{j+1}\left\{[\mathbf{Z}_{j+1}](\hat{\mathbf{y}}_j,[\boldsymbol{\theta}]) + \mathcal{T}_{\mathbf{u}_{j+1}}(\boldsymbol{\theta}) - \hat{\mathcal{P}}_{\mathbf{y}_{j+1}}(\boldsymbol{\theta})\right\}\end{aligned} \tag{5.17}$$

$$\mathcal{T}_{\mathbf{g}_{j+1}}(\mathcal{P}_{\mathbf{y}_j}(\boldsymbol{\theta}),\boldsymbol{\theta}) = \mathbf{A}_{j+1}\mathcal{T}_{\boldsymbol{\Gamma}_{j+1}}(\mathcal{P}_{\mathbf{y}_j}(\boldsymbol{\theta}),\boldsymbol{\theta}) \tag{5.18}$$

Equation (5.18) is the expression to be used as the constraint satisfaction problem. The interval Newton/Gauss-Seidel contractor is applied a number of times to (5.18) to reduce the overestimation.

5.5.4 Program implementation and Third Party Libraries

The methods explained before were implemented in purely in Mathematica. The built in interval type in Mathematica was used for the interval arithmetic operations. Furthermore, since there is no automatic differentiation library in Mathematica, the Taylor coefficients were propagated by defining the rules of automatic differentiation for all binary operations and intrinsic functions needed. For programming the Taylor Models, the rules of Taylor

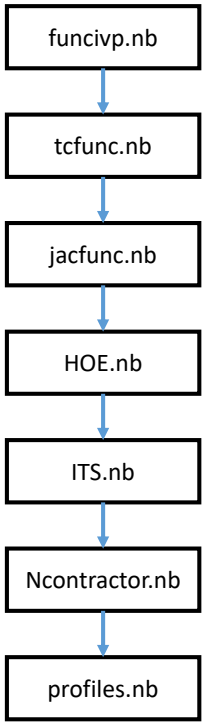


Figure 5.1: Diagram of the C++ programs to implement an ITS method with contractors

Model arithmetic for addition, multiplication and intrinsic functions were also defined in Mathematica.

Figure 5.1 represents the program used to implement the Taylor Models method with interval contractors NGS. At the top of the figure (file `funcivp.nb`), the mathematical model of the initial value problem is defined (note that the extension is nb because it was developed in Mathematica), the Taylor coefficients of the Taylor Models ($\mathcal{T}_{\mathbf{f}^{[i]}}(\mathcal{T}_{\mathbf{y}_j}(\boldsymbol{\theta}), \boldsymbol{\theta})$) and the Jacobian ($\mathcal{T}_{\mathbf{S}^y_{j+1}}(\boldsymbol{\theta})$) are computed next in `tcfunc.nb` and `jacfunc.nb`. An important difference in the Taylor Models program is that there is no need to compute $\mathcal{T}_{\mathbf{S}^\theta_{j+1}}(\boldsymbol{\theta})$ since the dependency of the parameters θ is already being accounted in the polynomial part of the Taylor Model. The information obtained in these programs is enough to compute the first and second stages which are the high order enclosure (`HOE.nb`) and the tighter enclosure (`ITS.nb`). The information from `jacfunc.nb` is also input to the contractor routine Newton/Gauss-Seidel, `Ncontractor,nb`. Finally, `profiles.nb` is in charge of iterating the program and arranging the information of the trajectories.

5.6 Numerical Case Studies

This section presents results on the developed algorithm using the seven numerical case studies from Chapter 3. First, the case studies are presented with the same amount of uncertainty in the initial conditions and/or system parameters as in Chapter 3 (Uncertain values 1) and then they are addressed using more uncertainty (Uncertain values 2). Each test problem subsection also presents the condition number of the problem across the uncertain parameters and time horizon considered. Only the condition numbers for uncertain values 2 are given, since the condition numbers for uncertain values 1 were already given in Capter 3. Problems with small condition numbers are well-conditioned and so small changes in the inputs yield small changes in the outputs. On the other hand, problems with a large condition number are ill-conditioned which means that small changes in the inputs can produce arbitrarily large changes in the outputs. Preliminary, one can expect that problems with large condition will be more challenging when trying to compute upper and lower bounds. The condition number of a differentiable function f can be calculated by:

$$\kappa = \frac{\|J(x)\|}{\|f(x)\|\|x\|} \tag{5.19}$$

where J is the Jacobian of f.

For the sake of completeness, the equations and tables for each problem presented in Chapter 3 are displayed in this chapter as well. The tables that include the parameters and uncertain values also contains extra rows showing the additional values of uncertainty to be considered. Below, each problem has a section in which a table reports the results on the test problems. The algorithmic implementation was developed in Mathematica 9 since it provides support for interval and symbolic arithmetic which is useful in Taylor models. The widths given corresponds to a selected time t_s. The computational time is not reported as the algorithm was developed in Mathematica first to prove the effectiveness of the new ideas. When applying a contractor iteration, the CPU time is roughly twice as much as the Taylor models alone. In Chapter 3 we provided computational times for a Taylor models method developed in C++ (VSPODE) using the uncertain values 1. In this chapter the

Table 5.1: Test problems used, number of state variables and parameters and relevance in dynamic optimisation

Problem	States	Parameters	Relevance
1. First order irreversible series reaction	2	2	Parameter estimation for model development
2. First order reversible series reaction	2	4	Parameter estimation for model development
3. Exothermic batch reactor	2	8	Guaranteed safe operation
4. Two-state bioreactor	2	6	Parameter estimation for model development
5. Three-state bioreactor	3	8	Parameter estimation for model development
6. Reactor separator model	6	9	Optimal control
7. Glucagon receptor model	5	22	Guaranteed safe operation

VSPODE method with order $q = 4$ has only been used in the uncertain values 2 to see how it compares with the Taylor Models method with Newton/Gauss-Seidel contractor. These two methods are simply identified as TM and TM-NGS, respectively.

5.6.1 First Order Irreversible Series Reaction

The first order irreversible series reaction problem has been simulated using Taylor Models (TM) and Taylor Models with the Newton/Gauss-Seidel (TM-NGS) contractor. The mathematical model is shown in Equations (5.20) and the system parameters and uncertain values are presented in Table 5.2. Two uncertainty levels have been used to address the effectiveness of the contractor in the Taylor model. The condition number for the uncertain values 1 can be found in Chapter 3 in Figure 3.2 and the condition number for the uncertain values 2 is given in Figure 5.2.

$$\begin{aligned} \dot{C}_A &= -k_1 C_A \\ \dot{C}_B &= k_1 C_A - k_2 C_B \end{aligned} \tag{5.20}$$

For the first order irreversible series reaction problem, the first two figures address the problem using the uncertain values 1 (Figures 5.3 and 5.4). It can be observed that both the method with and without contractor are able to tightly bound the trajectories for the whole time horizon. Figures 5.5 and 5.6 use much wider uncertain amounts and both of the

Table 5.2: Initial conditions and system parameters for the first order irreversible series reaction problem

Initial conditions and system parameters
$C_A(0) = 1$
$C_B(0) = 0$
Uncertain values 1
$k_1 = [4.5, 5.5]$
$k_2 = [0.2, 1.8]$
Uncertain values 2
$k_1 = [0.1, 9.9]$
$k_2 = [0.01, 1.99]$

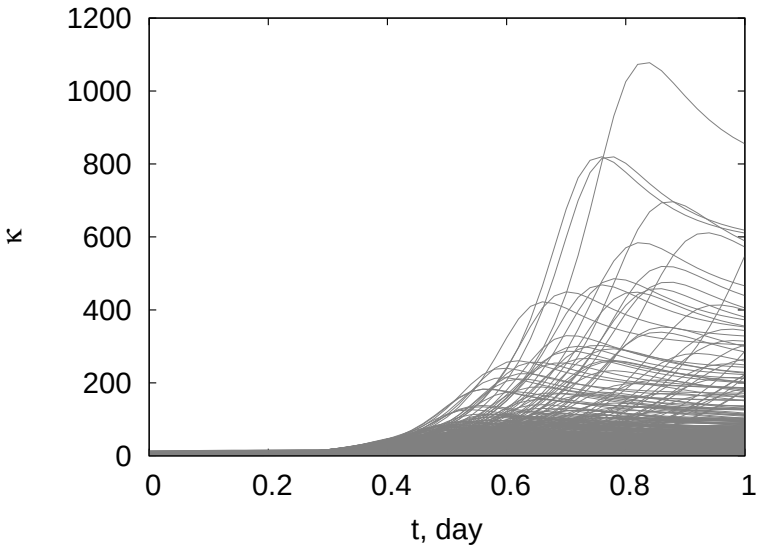


Figure 5.2: Condition number of first order irreversible series reaction across the time horizon and the parametric uncertainties (uncertain values 2)

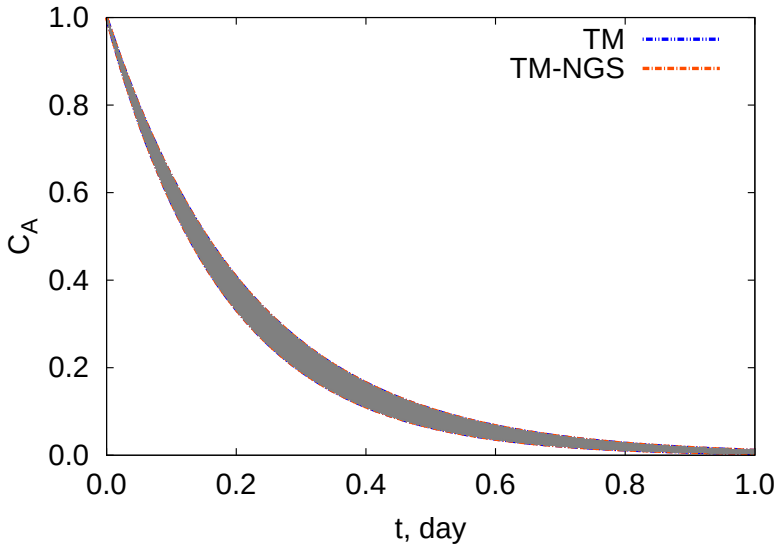


Figure 5.3: State variable C_A of first order irreversible series reaction using uncertain values 1

Table 5.3: Results on the implementation of interval contractors in the first order irreversible series reaction

Uncertainty	**Method**	**Width at $t_s = 1$ day**
1	TM	$w(C_A(t_s))$=0.007061
		$w(C_B(t_s))$=0.6327
	TM-NGS	$w(C_A(t_s))$=0.007062
		$w(C_B(t_s))$=0.6325
2	TM	$w(C_A(t_s))$=1.3909
		$w(C_B(t_s))$=5.06592
	TM-NGS	$w(C_A(t_s))$=1.1216
		$w(C_B(t_s))$=1.7627

methods produce more conservative bounds. The method with the NGS contractor provided tighter bounds in the state C_B. The values of the widths for uncertain values 1 and 2 and for both methods are given in Table 5.3. In the case of uncertain values 2, the widest method in the C_B state variable is the TM method (100%). The percentage that corresponds to the wideness of the TM-NGS method is $w_{TM-NGS}(C_B(t_s))/w_{TM}(C_B(t_s)) \times 100 = 35\%$

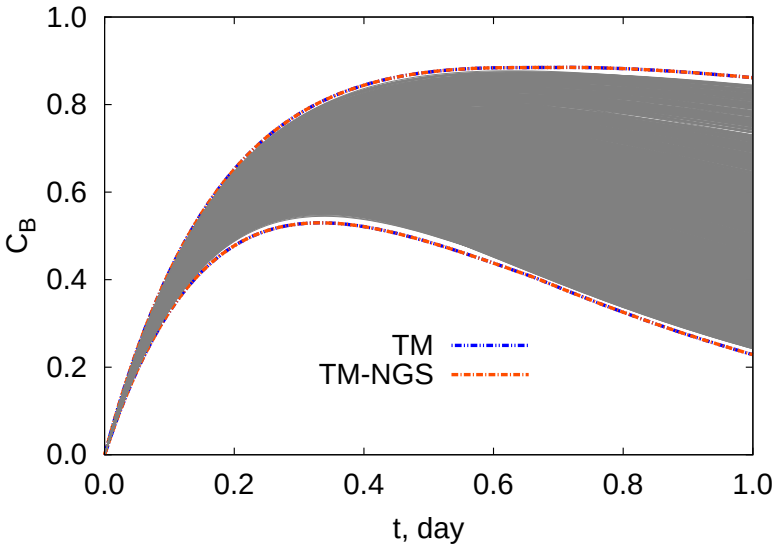


Figure 5.4: State variable C_B of first order irreversible series reaction using uncertain values 1

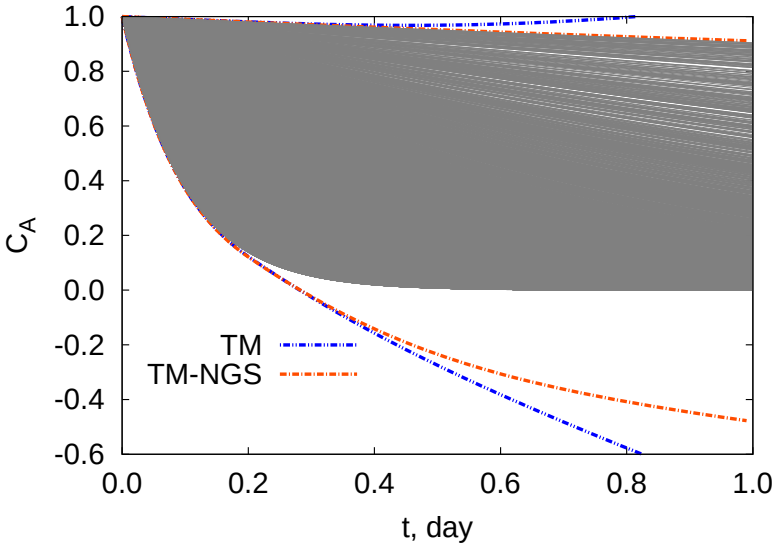


Figure 5.5: State variable C_A of first order irreversible series reaction using uncertain values 2. The TM bounds were constructed using the VSPODE method with $q = 4$

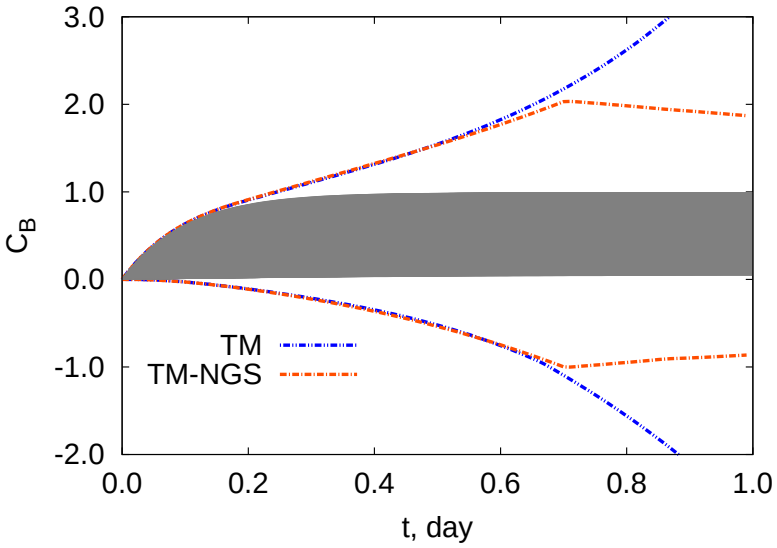


Figure 5.6: State variable C_B of first order irreversible series reaction using uncertain values 2. The TM bounds were constructed using the VSPODE method with $q = 4$

Table 5.4: Initial conditions and system parameters for the first order reversible series reaction problem

Initial conditions and system parameters
$C_A(0) = 1$
$C_B(0) = 0$
$k_{-1} = 2$
$k_{-2} = 20$
Uncertain values 1
$k_1 = [2, 6]$
$k_{-1} = [1, 3]$
Uncertain values 2
$k_1 = [0, 15]$
$k_{-1} = [0, 10]$

5.6.2 First Order Reversible Series Reaction

The first order reversible series reaction has been addressed using two levels of uncertainty and the two methods TM and TM-NGS. The mathematical model of the problem is presented in (5.21) and the system paramaters and the uncertain values 1 and 2 are displayed in Table 5.4. The condition number for the uncertain values 1 can be found in Chapter 3 in Figure 3.5. This subsection presents only the condition number using uncertain values 2 in Figure 5.7. This condition number is much larger when uncertain values 2 are used, so a bigger challenge is expected when bounding its trajectories.

$$\begin{aligned} \dot{C}_A &= -k_1 C_A + k_{-1} C_B \\ \dot{C}_B &= k_1 C_A - (k_{-1} + k_2) C_B + k_{-2}(1 - C_A - C_B) \end{aligned} \tag{5.21}$$

As expected the two methods are able to tightly bound the problem with uncertain values 1 (Figures 5.8 and 5.9). However, Figures 5.10 and 5.11 show that much more conservative bounds are obtained when the uncertainty increases to uncertain values 2. Table 5.5 reports the widths using both methods and both uncertainty levels. Since both methods yielded similar tightness in uncertain values 1, the widths of uncertain values 2 are compared. The TM method turned out to be the one with largest width in C_A state variable. The width of the TM method is taken as the 100% of width and so the TM-NGS method corresponds to $w_{TM-NGS}(C_A(t_s))/w_{TM}(C_A(t_s)) \times 100 = 65\%$.

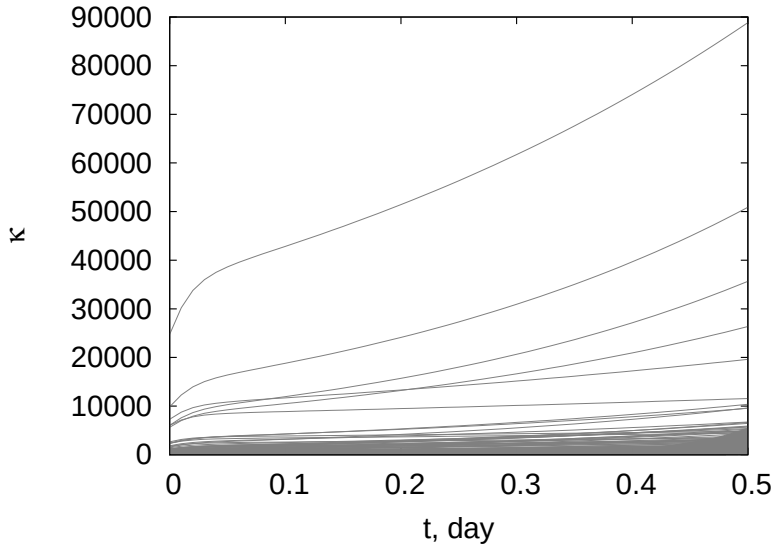


Figure 5.7: Condition number of first order reversible series reaction across the time horizon and the parametric uncertainties (uncertain values 2)

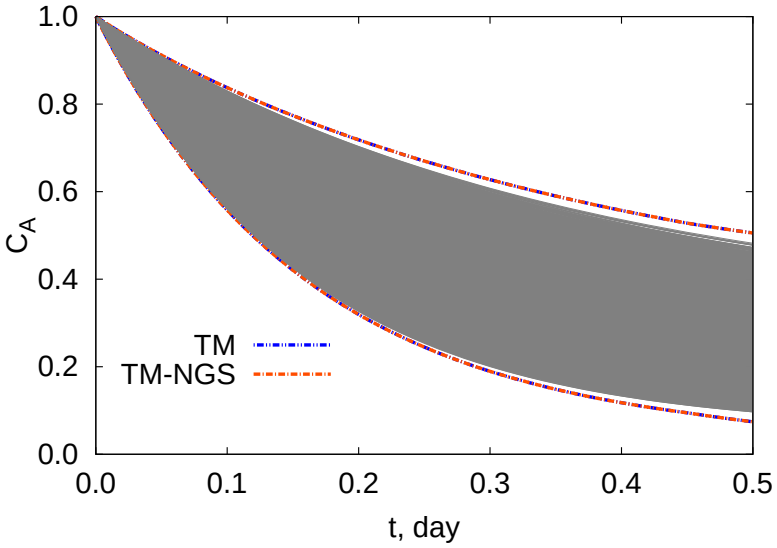


Figure 5.8: State variable C_A of first order reversible series reaction using uncertain values 1

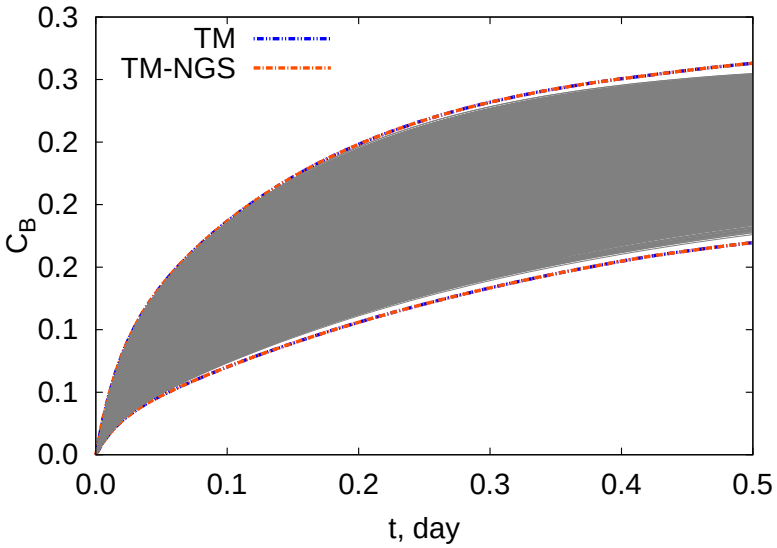


Figure 5.9: State variable C_B of first order reversible series reaction using uncertain values 1

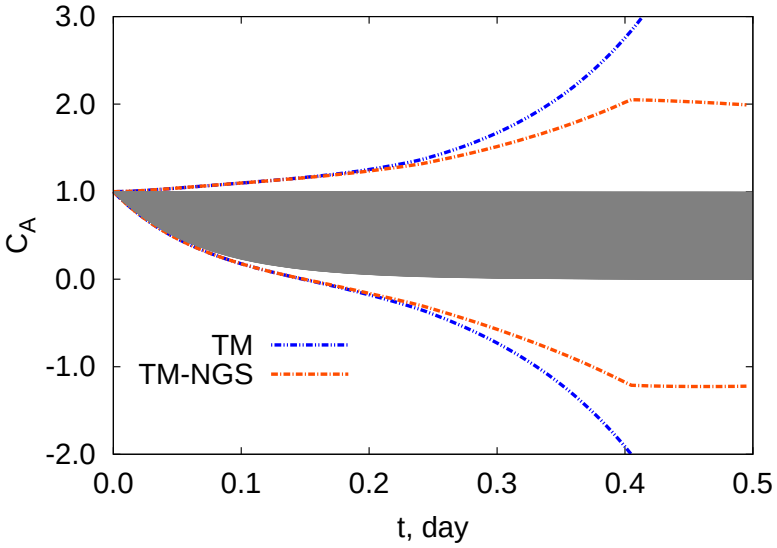


Figure 5.10: State variable C_A of first order reversible series reaction using uncertain values 2. The TM bounds were constructed using the VSPODE method with $q = 4$

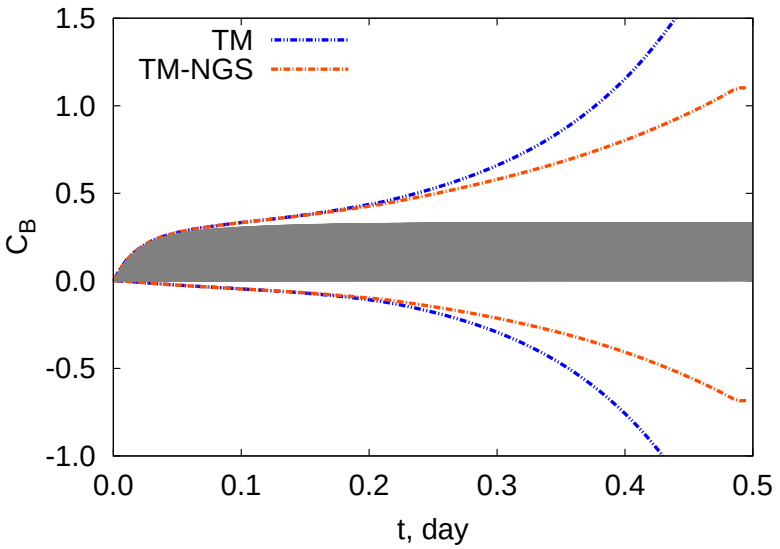


Figure 5.11: State variable C_B of first order reversible series reaction using uncertain values 2. The TM bounds were constructed using the VSPODE method with $q = 4$

Table 5.5: Results on the implementation of interval contractors in the first order reversible series reaction

Uncertainty	**Method**	**Width at $t_s = 0.5$ day**
1	TM	$w(C_A(t_s))$=0.4054
		$w(C_B(t_s))$=0.1436
	TM-NGS	$w(C_A(t_s))$=0.4054
		$w(C_B(t_s))$=0.1436
2	TM	$w(C_A(t_s))$=4.9834
		$w(C_B(t_s))$=1.8936
	TM-NGS	$w(C_A(t_s))$=3.2126
		$w(C_B(t_s))$=1.7858

Table 5.6: Initial conditions and system parameters for the exothermic batch reactor problem

Initial conditions and system parameters
$x(0) = 0$
$k_0 = 0.022$
$V = 0.1$
$C_p = 60$
$E_a = 6000$
$R = 8.314$
$\Delta H_R = -140000$
$UA = 3$
$C_{A0} = 10$
Uncertain values 1
$T(0) = [310, 410]$
$T_a = [290, 310]$
Uncertain values 2
$T(0) = [285, 435]$
$T_a = [280, 320]$

5.6.3 Exothermic Batch Reaction

The mathematical model of the exothermic batch reactor is given in (5.22) and its system parameters and uncertain values 1 and 2 are presented in Table 5.6. Only the condition number of uncertain values 2 is given in this chapter since the condition number using uncertain values 1 was given in Chapter 3 in Figure 3.8. Figure 5.12 shows the condition number with uncertain values 2. In this particular problem, changing the uncertainty from uncertain values 1 to 2 does not seem to have a big effect. The only difference observed is that the condition number starts growing large earlier in the simulation time in the case of uncertain values 2.

$$\begin{aligned} \dot{x} &= k_0(1-x)\mathrm{e}^{-\frac{E_a}{RT}} \\ \dot{T} &= \frac{UA}{C_{A0}VC_p}(T_a - T) - \frac{\Delta H_R}{C_p}k_0(1-x)\mathrm{e}^{-\frac{E_a}{RT}} \end{aligned} \tag{5.22}$$

The main comparison criteria in this test problem is the width of both of the state variables when using uncertain values 2. The tightness is very similar when using uncertain values 1. The Figures 5.13 and 5.14 show the bounds of both methods for uncertain values 1. According to Table 5.7, the widest method when using the uncertain values 2 is the TM method. Therefore, the width of the TM method is taken as the 100% and the corresponding

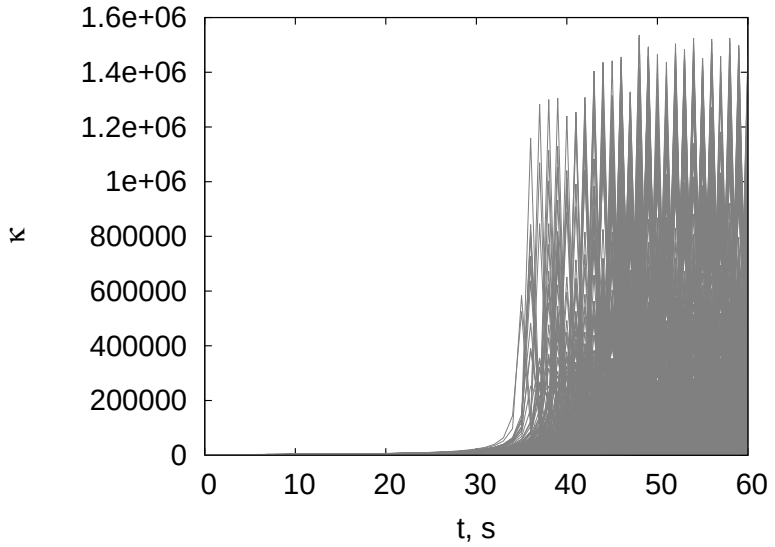


Figure 5.12: Condition number of exothermic batch reaction across the time horizon and the parametric uncertainties (uncertain values 2)

Table 5.7: Results on the implementation of interval contractors in the exothermic batch reaction

Uncertainty	**Method**	**Width at $t_s = 60$ s**
1	TM	$w(x(t_s))$=0.07348
		$w(T(t_s))$=59.3314
	TM-NGS	$w(x(t_s))$=0.06686
		$w(T(t_s))$=59.1925
2	TM	$w(x(t_s))$=0.1163
		$w(T(t_s))$=109.2209
	TM-NGS	$w(x(t_s))$=0.1076
		$w(T(t_s))$=108.8168

percentages of this width for both state variables is $w_{TM-NGS}(x(t_s))/w_{TM}(x(t_s)) \times 100 = 93\%$ for x and $w_{TM-NGS}(T(t_s))/w_{TM}(T(t_s)) \times 100 = 99\%$ for T. The Figures 5.15 and 5.16 show the bounds of both methods for uncertain values 2.

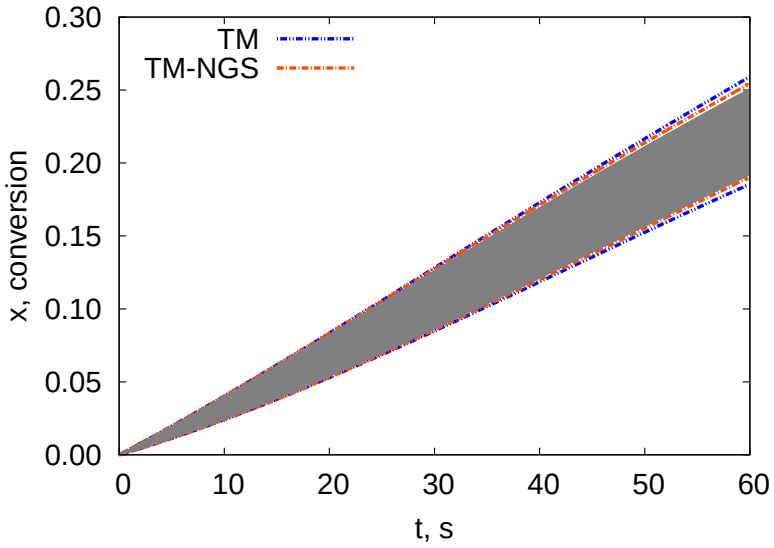


Figure 5.13: State variable x of exothermic batch reaction using uncertain values 1

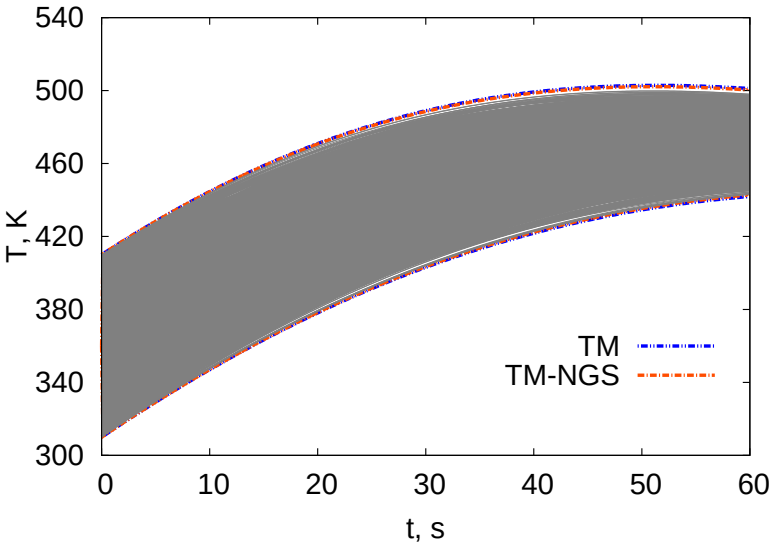


Figure 5.14: State variable T of exothermic batch reaction using uncertain values 1

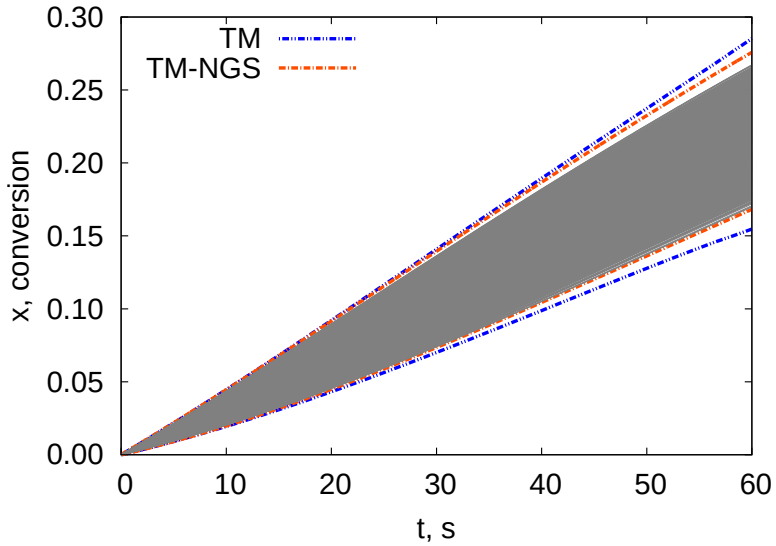


Figure 5.15: State variable x of exothermic batch reaction using uncertain values 2. The TM bounds were constructed using the VSPODE method with $q = 4$

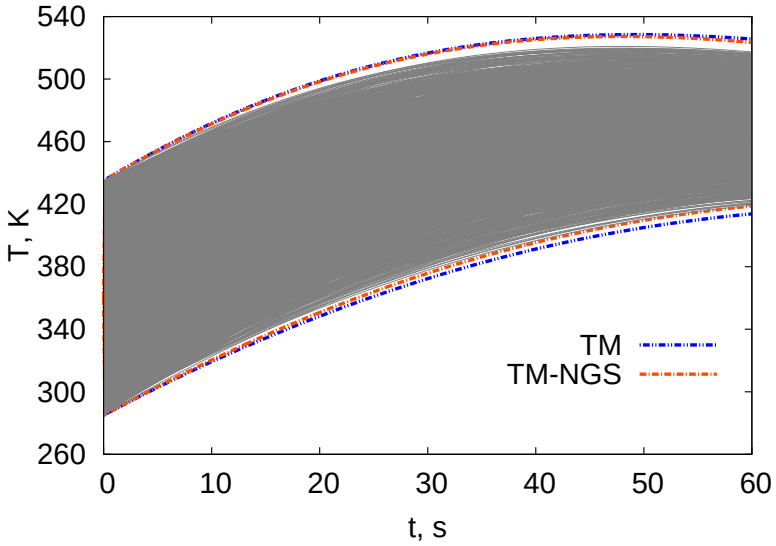


Figure 5.16: State variable T of exothermic batch reaction using uncertain values 2. The TM bounds were constructed using the VSPODE method with $q = 4$

Table 5.8: Initial conditions and system parameters for the two-state bioreactor problem

Initial conditions and system parameters
$S(0) = 0.8$
$\alpha = 0.5$
$k = 10.53$
$D = 0.36$
$S_f = 5.7$
$K_s = 7.0$
$k_0 = 0.022$
Uncertain values 1
$X(0) = [0.80, 0.85]$
$\mu_{\max} = [1.1, 1.2]$

5.6.4 Two-state Bioreactor

In this case only one set of uncertain values was tested, since the current implementation is only able to address low levels of uncertainties for these problems. Since it was developed from scratch, not all the VSPODE heuristics have been included. Note that bounds for the same level of uncertainty using the VSPODE method were provided in Chapter 3. The mathematical model is presented in (5.23) and the system parameters and uncertain values are given in Table 5.8. The condition number for this level of uncertainty was given also in Chapter 3 in Figure 3.11 where it was observed that it has large values (almost asymptotic) half way the integration time and at the end.

$$\begin{aligned} \dot{X} &= (\mu - \alpha D)X \\ \dot{S} &= D(S_f - S) - k\mu X \\ \mu &= \frac{\mu_{\max} S}{K_s + S} \end{aligned} \tag{5.23}$$

The method using contractors provided tighter bounds than the implementation using only Taylor models. the trajectories of both methods can be seen in Figures 5.17 and 5.18. The width of the more conservative method is taken as the reference maximum width to calculate the percentage that corresponds to the other method. The widths can be found in Table 5.9. In this case, the width of the TM method represents the 100%. The percentages of both state variables using the TM-NGS method are calculated as $w_{TM-NGS}(X(t_s))/w_{TM}(X(t_s)) \times 100 = 22\%$ for X and $w_{TM-NGS}(S(t_s))/w_{TM}(S(t_s)) \times 100 = 41\%$ for S.

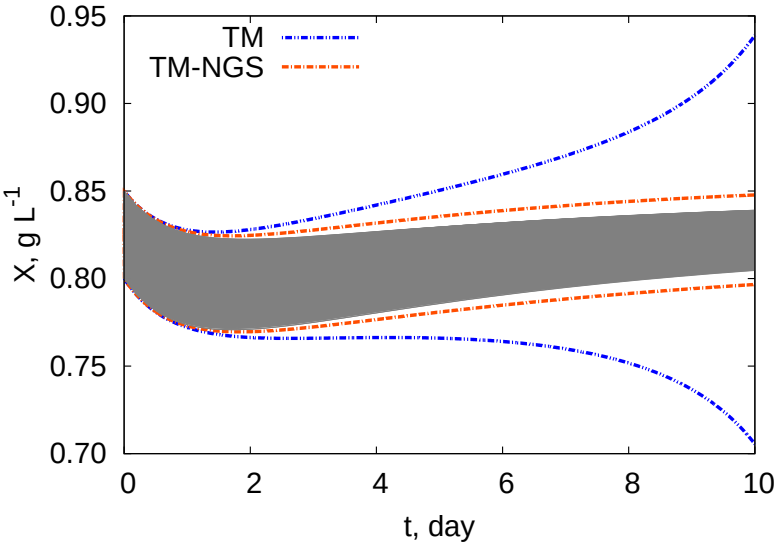


Figure 5.17: State variable X of two-state bioreactor using uncertain values 1

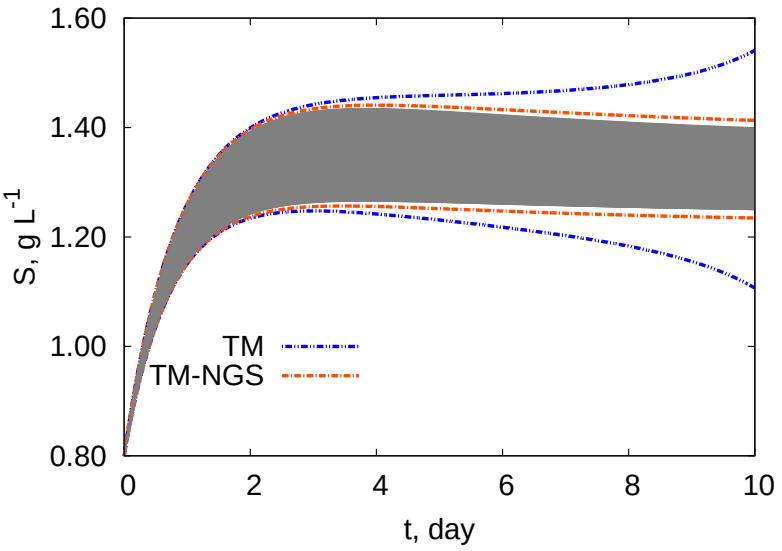


Figure 5.18: State variable S of two-state bioreactor using uncertain values 1

Table 5.9: Results on the implementation of interval contractors in the two-state bioreactor

Uncertainty	**Method**	**Width at $t_s = 10$ day**
1	TM	$w(X(t_s))$=0.2334
		$w(S(t_s))$=0.4345
	TM-NGS	$w(X(t_s))$=0.05113
		$w(S(t_s))$=0.1784

Table 5.10: Initial conditions and system parameters for the three-state bioreactor problem

Initial conditions and system parameters
$x_2(0) = 5.0$
$x_3(0) = 15.0$
$D = 0.202$, $x_{2f} = 20$
$Y = 0.4$
$\beta = 0.2$
$x_{3m} = 50$
$\alpha = 0.5$
Uncertain values 1
$x_1(0) = [6.45, 6.55]$
$\mu_{\max} = [0.46, 0.47]$
$k_s = [1.05, 1.1]$
Uncertain values 2
$x_1(0) = [6.4, 6.6]$
$\mu_{\max} = [0.45, 0.48]$
$k_s = [1.03, 1.12]$

5.6.5 Three-state Bioreactor

The three-state bioreactor problem has been simulated using Taylor Models (TM) and Taylor Models with the Newton/Gauss-Seidel (TM-NGS) contractor. The mathematical model is shown in Equations (5.24) and the system parameters and uncertain values are presented in Table 5.10. Two uncertainty levels have been used to address the effectiveness of the contractor in the Taylor model. The condition number for the uncertain values 1 can be found in Chapter 3 in Figure 3.14 and the condition number for the uncertain values 2 is given in Figure 5.19.

$$\begin{aligned} \dot{x}_1 &= (\mu - D)x_1 \\ \dot{x}_2 &= D(x_{2f} - x_2) - \frac{\mu x_1}{Y} \\ \dot{x}_3 &= Dx_3 + (\alpha\mu + \beta)x_1 \\ \mu &= \frac{\mu_{\max}[1 - (\frac{x_3}{x_{3m}})]x_2}{k_s + x_2} \end{aligned} \tag{5.24}$$

As expected, the two methods are able to tightly bound the problem with uncertain values 1 (Figures 5.20 and 5.21). However, Figures 5.22 and 5.23 show that much more conservative bounds are obtained when the uncertainty increases to uncertain values 2. Table 5.11 reports the widths using both methods and both uncertainty levels. Since both methods yielded similar tightness in uncertain values 1, the widths of uncertain values 2 are compared. The

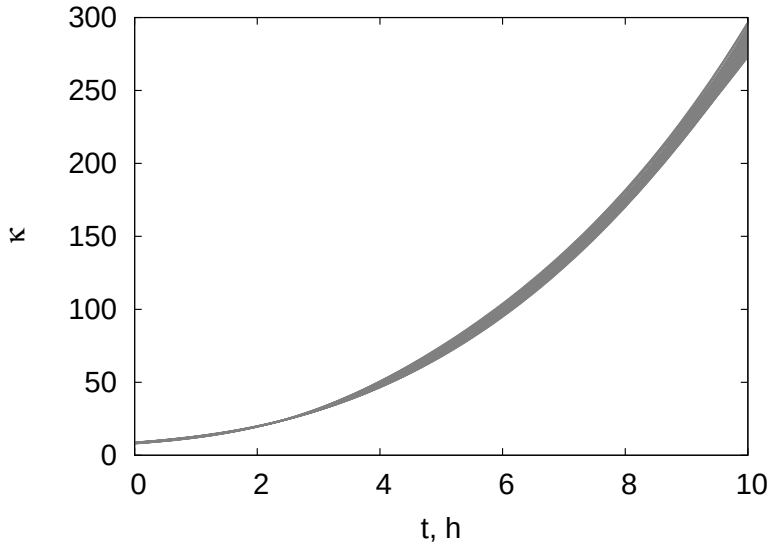


Figure 5.19: Condition number of three-state bioreactor across the time horizon and the parametric uncertainties (uncertain values 2)

Table 5.11: Results on the implementation of interval contractors in the three-state bioreactor

Uncertainty	**Method**	**Width at $t_s = 10$ day**
1	TM	$w(x_1(t_s))$=0.1478
		$w(x_2(t_s))$=0.2813
	TM-NGS	$w(x_1(t_s))$=0.1477
		$w(x_2(t_s))$=0.2812
2	TM	$w(x_1(t_s))$=0.2993
		$w(x_2(t_s))$=0.6756
	TM-NGS	$w(x_1(t_s))$=0.4629
		$w(x_2(t_s))$=0.8246

TM method using uncertain values 2 (VPSODE) turned out to provide better bounds than the contractor methods. It is worth mentioning that the method with contractors is in early stage of development and does not include all of the VSPODE heuristics. The width of the TM-NGS method is taken as the 100% of width and so the TM method corresponds to $w_{TM}(x_1(t_s))/w_{TM-NGS}(x_1(t_s)) \times 100 = 65\%$ for x_1 and $w_{TM}(x_2(t_s))/w_{TM-NGS}(x_2(t_s)) \times 100 = 82\%$ for x_2.

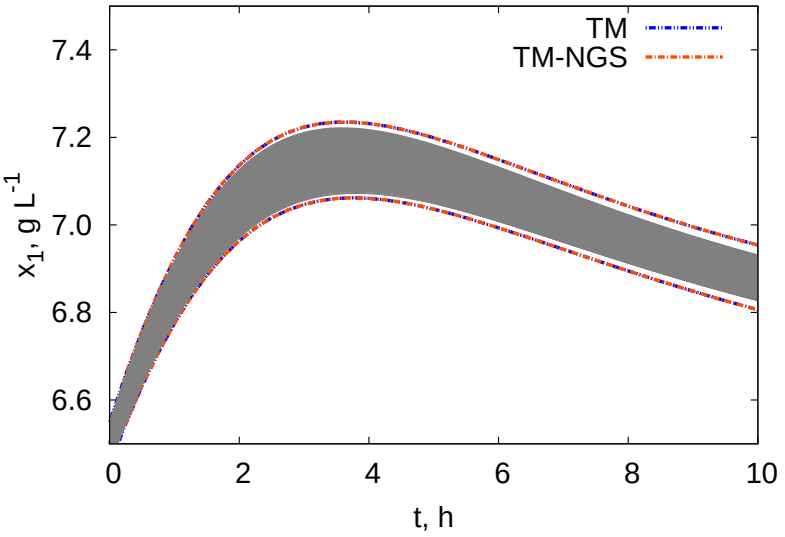


Figure 5.20: State variable x_1 of three-state bioreactor using uncertain values 1

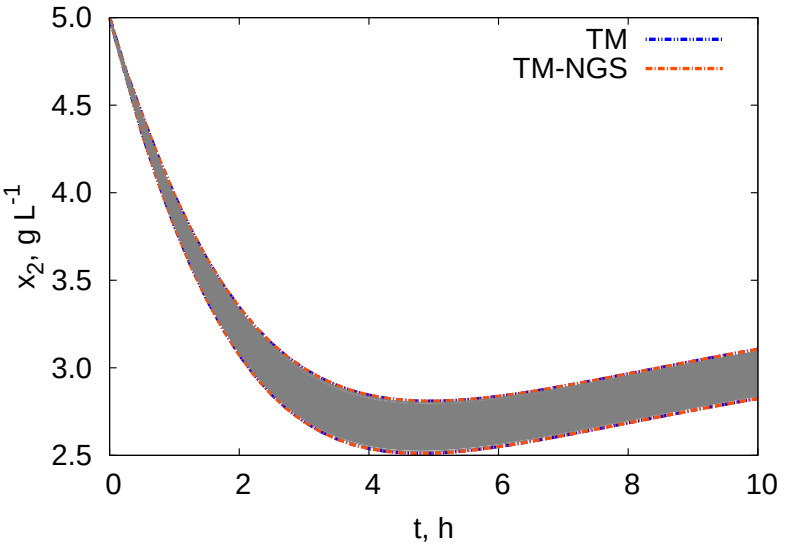


Figure 5.21: State variable x_2 of three-state bioreactor using uncertain values 1

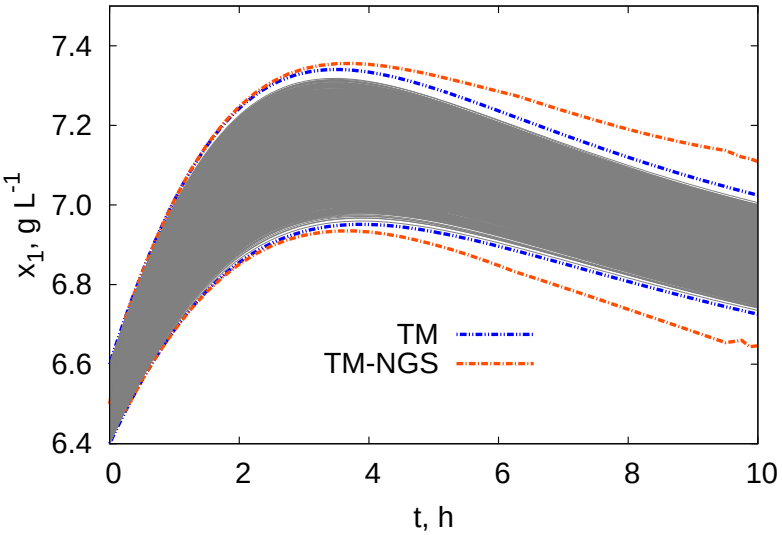


Figure 5.22: State variable x_1 of three-state bioreactor using uncertain values 2. The TM bounds were constructed using the VSPODE method with $q = 4$

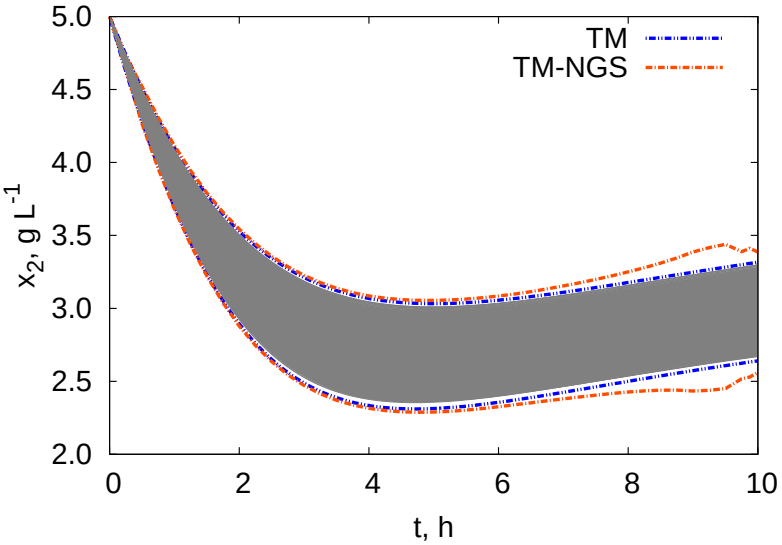


Figure 5.23: State variable x_2 of three-state bioreactor using uncertain values 2. The TM bounds were constructed using the VSPODE method with $q = 4$

5.6.6 Reactor Separator Model

In the reactor separator model,only one set of uncertain values was tested since the current implementation is only able to address low levels of uncertainties for this problems. Since it was developed from scratch, not all the VSPODE heuristics have been included. Note that bounds for the same level of uncertainty using the VSPODE method were provided in Chapter 3. The mathematical model is presented in (5.25) and the system parameters and uncertain values are given in Table 5.12. The condition number for this level of uncertainty was given also in Chapter 3 in Figure 3.17 where it was observed that the problem is ill-conditioned in the second half of the integration time as it suddenly increases around half way of the integration.

$$\begin{aligned}
\dot{x_1} &= \frac{F+B}{H}(x_F - x_1) + kx_1(1-x_1) \\
\dot{x_2} &= (L+F+B)x_3 - Bx_2 - Vy_2 \\
\dot{x_3} &= (L+F+B)(x_4 - x_3) + V(y_2 - y_3) \\
\dot{x_4} &= (F+B)x_1 + Lx_5 - (L+F+B)x_4 + V(y_3 - y_4) \\
\dot{x_5} &= L(x_6 - x_5) + V(y_4 - y_5) \\
\dot{x_6} &= -(L+D)x_6 + Vy_5 \\
x_F &= \frac{Fx_{F0} + Bx_2}{F+B} \\
y_i &= \frac{\alpha x_i}{1+(\alpha-1)x_i} \;\; i = 2\ldots 5
\end{aligned} \tag{5.25}$$

The method using contractors provided tighter bounds than the implementation using only Taylor models. The trajectories of both methods can be seen in Figures 5.24 and 5.25. The width of the more conservative method is taken as the reference maximum width to calculate the percentage that corresponds to the other method. The widths can be found in Table 5.13. In this case, the width of the TM method represents the 100%. The percentages of both state variables using the TM-NGS method are calculated as $w_{TM-NGS}(x_1(t_s))/w_{TM}(x_1(t_s)) \times 100 = 25\%$ for x_1 and $w_{TM-NGS}(x_6(t_s))/w_{TM}(x_6(t_s)) \times 100 = 10\%$ for x_2.

Table 5.12: Initial conditions and system parameters for the reactor separator model

Initial conditions and system parameters
$x_1(0) = 0.5$
$x_2(0) = 0.0$
$x_3(0) = 0.0$
$x_4(0) = 0.0$
$k = 0.06$
$x_{F0} = 0$
$D = 1$
$H = 63.33$
$\alpha = 7.5$
$B = 1.2$
$L = 1.704$
$F = 1.0$
Uncertain values 1
$x_5(0) = [0.0, 0.08]$
$x_6(0) = [0.0, 0.16]$

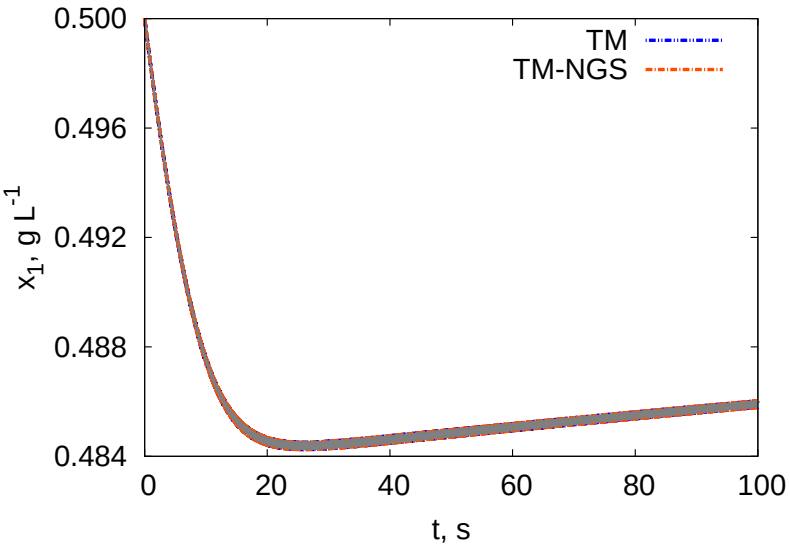


Figure 5.24: State variable x_1 of reactor separator model using uncertain values 1

Table 5.13: Results on the implementation of interval contractors in the reactor separator model

Uncertainty	**Method**	**Width at $t_s = 100$ s**
1	TM	$w(x_1(t_s))$=0.00007711
		$w(x_6(t_s))$=0.1939
	TM-NGS	$w(x_1(t_s))$=0.00001953
		$w(x_6(t_s))$=0.01927

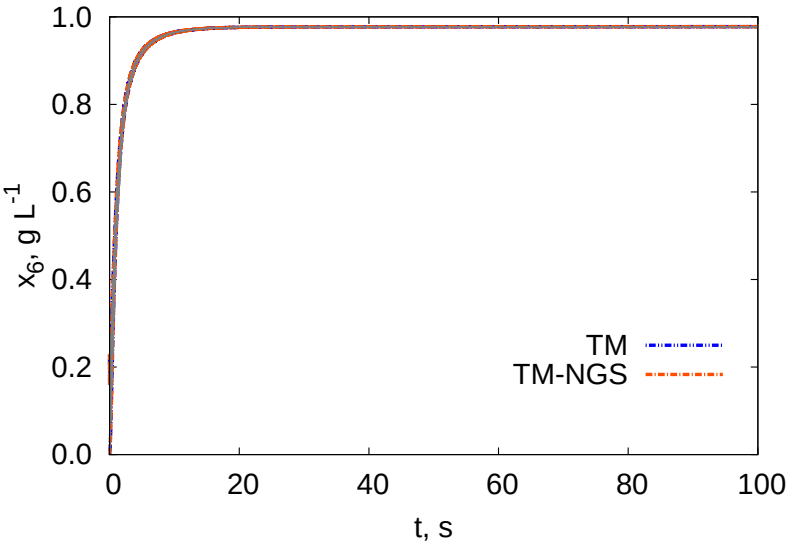


Figure 5.25: State variable x_6 of reactor separator model using uncertain values 1

5.6.7 Glucagon Receptor Model

The glucagon receptor model has been addressed using two levels of uncertainty and the two methods TM and TM-NGS. The mathematical model of the problem is presented in (5.26) and the system parameters and the uncertain values 1 and 2 are displayed in Table 5.14. The condition number for the uncertain values 1 can be found in Chapter 3 in Figure 3.20. This subsection presents the condition number using uncertain values 2, which can be seen in Figure 5.26. This condition number is much larger when uncertain values 2 are used, so a bigger challenge is expected when bounding the glucagon receptor model trajectories.

$$
\begin{aligned}
\dot{R}_r &= k_{-1} L R_u - L k_1 R_r - k_s R_r + k_r R_s \\
\dot{R}_s &= k_{sp} L R_p + G_i K_{2s} L R_u + k_s (L R_u + R_r) - k_r R_s \\
\dot{G}_i &= -G_i K_{23} L R_u + G_* \left(k_h + \frac{Ca\, k_{Gdeg,Cal}}{K_{Gdeg,Cal} + G_*} + \frac{PLC_*\, k_{Gdeg,PLC}}{K_{Gdeg,PLC} + G_*} \right) \\
\dot{LR_p} &= -k_{sp} L R_p + k_p \left(1 + \frac{A_0}{1 + B_1 G_*^{-n_1}} \right) \left(\frac{L R_u}{L R_u + B_2} \right) \\
P\dot{L}C_* &= k_{PC} G_* - \frac{PLC_* k_{PC,deg}}{K_{PC,deg} + PLC_*} \\
G_* &= G_0 - G_i \\
R_0 &= R_r + R_s + L R_u + L R_p
\end{aligned}
\tag{5.26}
$$

For the glucagon receptor model, the first two figures address the problem using the uncertain values 1 (Figures 5.27 and 5.28). It can be observed that both the method with and without contractor are able to tightly bound the trajectories for the whole time horizon. Figures 5.29 and 5.30 use much wider uncertain amounts and both of the methods produce more conservative bounds. The values for the widths for uncertain values 1 and 2 and for method TM-NGS are given in Table 5.15. The TM method is not included in the case of uncertain values 2 (VSPODE) because it had to stop early due to overestimation. Only the TM-NGS method was able to complete the whole integration time with uncertain values 2.

Table 5.14: Initial conditions and system parameters for the glucagon receptor model

Initial conditions and system parameters
$k_1 = 100$
$k_s = 5.2 \times 10^{-3}$
$k_{sp} = k_s$
$K_{2s} = 2 \times 10^{-8}$
$k_r = 4 \times 10^{-3}$
$K_{23} = 1 \times 10^{-7}$
$k_h = 2 \times 10^{-1}$
$k_{Gdeg,Cal} = 1.47 \times 10^3$
$K_{Gdeg,Cal} = 3.54 \times 10^1$
$k_{Gdeg,PLC} = 2.19 \times 10^3$
$K_{Gdeg,PLC} = 5.7$
$k_p = 6.5 \times 10^4$
$A_0 = 3$
$k_{-1} = 10$
$n_1 = 1$
$B_2 = 1 \times 10^6$
$R_0 = 5.5 \times 10^4$
$G_0 = 1 \times 10^5$
$k_{pc} = 6.06 \times 10^{-4}$
$k_{PC,deg} = 2.82 \times 10^{-1}$
$K_{PC,deg} = 2.55 \times 10^{-1}$
Uncertain values 1
$B_1 = [98.8, 102.2]$
Uncertain values 2
$B_1 = [70, 130]$

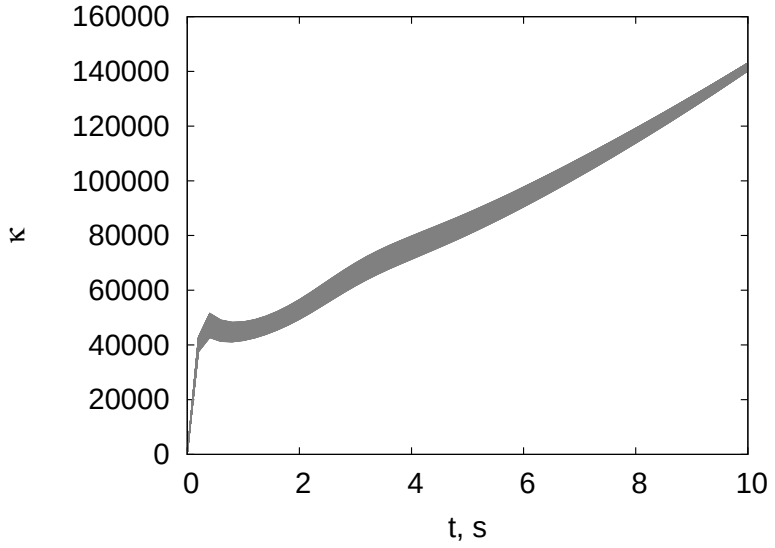


Figure 5.26: Condition number of Glucagon receptor model across the time horizon and the parametric uncertainties (uncertain values 2)

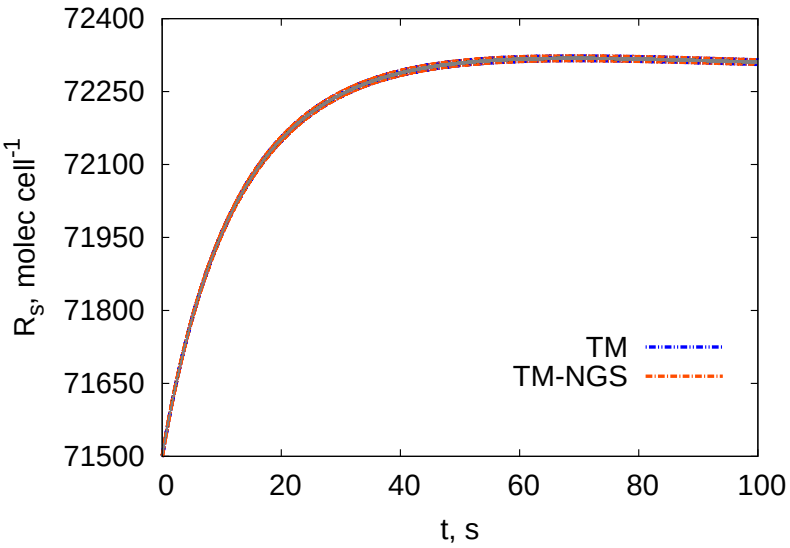


Figure 5.27: State variable R_s of Glucagon receptor model using uncertain values 1

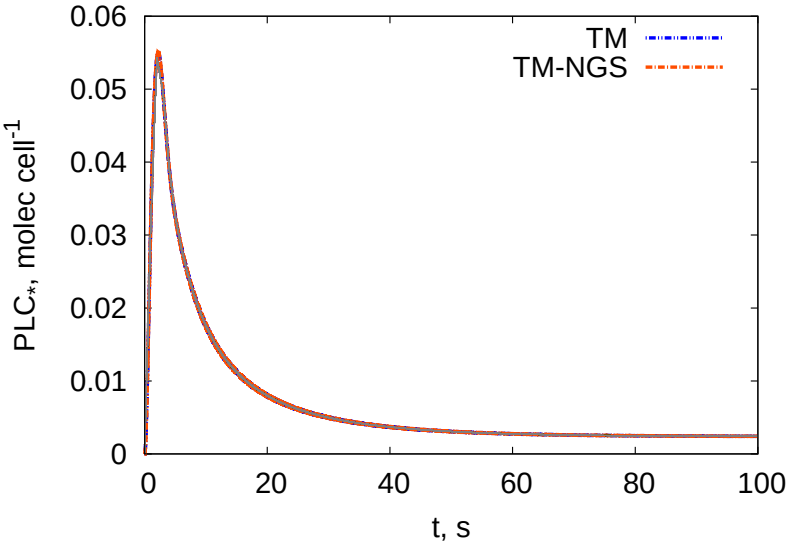


Figure 5.28: State variable PLC_* of Glucagon receptor model using uncertain values 1

Table 5.15: Results on the implementation of interval contractors in the Glucagon receptor model

Uncertainty	**Method**	**Width at $t_s = 100$ s**
1	TM	$w(R_s(t_s))$=9.1469
		$w(PLC_*(t_s))$=0.00003334
	TM-NGS	$w(R_s(t_s))$=9.1469
		$w(PLC_*(t_s))$=0.00003334
2	TM	$w(R_s(t_s))$=37.6411
		$w(PLC_*(t_s))$=0.008557
	TM-NGS	$w(R_s(t_s))$=34.2882
		$w(PLC_*(t_s))$=0.005041

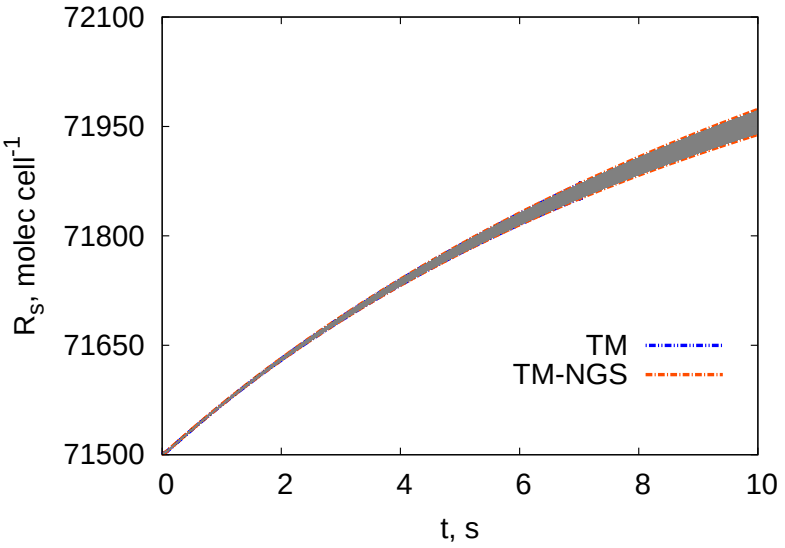


Figure 5.29: State variable R_s of Glucagon receptor model using uncertain values 2. The TM bounds were constructed using the VSPODE method with $q = 4$

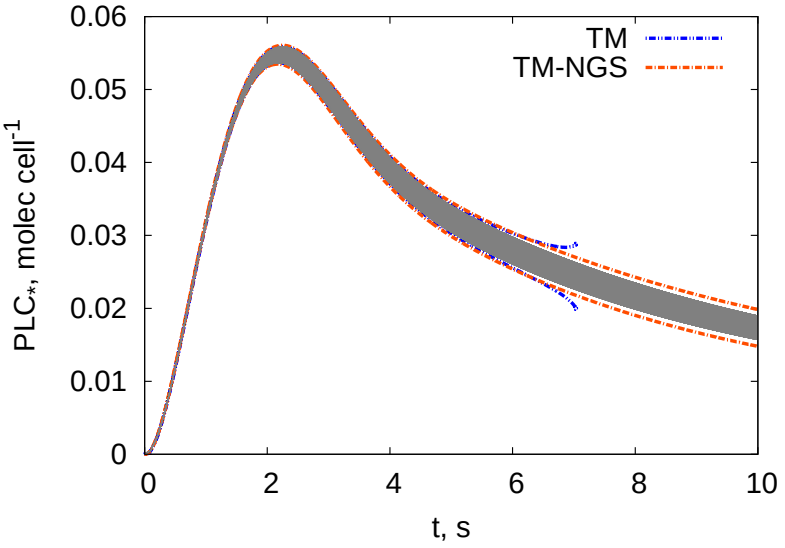


Figure 5.30: State variable PLC_* of Glucagon receptor model using uncertain values 2. The TM bounds were constructed using the VSPODE method with $q = 4$

5.7 Conclusions

This chapter has demonstrated the application of a fixed-point interval contractor in a Taylor models method for verified integration. Seven case studies have been used to test the capabilities of the method. The method was effective in reducing the overestimation in problems with two levels of uncertain values.

The condition numbers for all the problems were also presented. According to the two levels of uncertainty used in the test problems, when the uncertainty in the system parameters or initial conditions is increased, the condition number is increased. This gives us an idea how how challenging it is to bound, a certain problem.

The VSPODE method was used in the higher uncertain values 2 in the test problems. These simulations gave us an idea of how much uncertainty is required to make a Taylor Models method fail. Furthermore, the simulations with Taylor Models and interval contractors resulted to be tighter most of the time. This is a promising finding since it means that implementing the contractor methodology in a faster language can lead to improvements in the bounding capability of Taylor Models when addressing problems with large uncertainties. Furthermore, a similar methodology as in Chapter 4 can be applied to Taylor Models to add the constraint propagation capability to this approach.

Regarding the computational expense, the CPU times are approximately twice as the Taylor models method alone. The use of the method is justified according to how much overestimation is achieved. In our experience, the method was more effective in the sixth and seventh case studies as the overestimation reduction is significant. As discussed in Chapter 3, an alternative to reduce the CPU time is to include an heuristic to selectively perform the contractor iteration when it is really needed and not at every time step.

Finally, the use of this algorithm is a promising alternative for global optimisation for dynamic systems since larger uncertain values can be included in the problems. Also, the use of other Taylor model methods such as the ellipsoidal remainder term from Houska

et al. [24] is a promising alternative for applying interval contractors. It is also advisable the implementation of these algorithms in a compiled language such as C++.

Chapter 6

Global Optimisation for Dynamic Systems with Overestimation Reduction

Engineers seek optimal solutions in designing systems, but a crucial element is to ensure bounded performance. For example chemical reactors are often very heavy energy users, so it is important to find designs that minimise energy use, but the solution must be within strict safety limits. Currently, the deterministic solution of dynamic systems to global optimality can only be addressed for small problems. The solution of the ordinary differential equation (ODE) systems in a verified way are only able to address low dimensional problems mainly because the integration has to be stopped early due to the overestimation generated by the verified method. Chemical engineering researchers have used a range of techniques to tackle this problem using ways of finding tight over/under-estimators. In our work, a verified solver that constructs upper and lower bounds on the dynamic variables of initial value problems (IVP) for ODEs is used in a dynamic global optimisation method (sequential approach). Particular attention is paid to the reduction of the overestimation by means of interval contractors. The solver is used to provide guaranteed bounds on the objective function and on the first order sensitivity equations in a branch and bound framework. Uncertainty can be introduced in the dynamic constraints of the dynamic optimisation problem and therefore it is possible to account for it in a guaranteed way. This chapter will review research work

in chemical engineering for such problems and present results of work we have undertaken using interval methods. The chapter shows three examples from process engineering.

6.1 Introduction

In chemical engineering, dynamic processes arise in many applications and often an optimal trajectory is sought. For example, we need the control path for optimal resource consumption in changes of product grade on polymer plants, estimate dynamic states for process control as well as in model building applications [19, 75]. There are also important applications such as the design of batch systems in chemical engineering, including dynamic rectors and a system with a reactor, heat exchanger, separator and distillation column [8]. Furthermore, simultaneous dynamic optimisation is needed for the calculation of optimal transient trajectories for a polymerisation process [21], dynamic optimisation of distillation columns with particular focus on equilibrium constraints [59], computation of optimal time trajectories for the control of the different flows in a simulated moving-bed process [28], and a logic-based solution approach applied to a multistage batch distillation process of a ternary mixture [51].

There are many problems that require guaranteed bounded performance along the whole trajectory often because of safety critical and environmental limits. In these applications, it is not admissible to allow a certain variable to go beyond some prescribed limits. For example, the content of a certain compound in a stream is not allowed to be present in a higher concentration than the level permitted by the environmental regulator who might otherwise oblige the plant to shut down. The engineer has to make sure that this concentration is within the admissible limits at all times, however, it must be done without compromising too much, the cost of the plant operation and the qualities of all products. Safety critical plants require that the variables of interest, for example temperature or pressure of plant equipment, be within limits for the whole time of operation.

Obtaining the optimal performance is not an easy task; since dynamic models in chemical engineering often exhibit non-convexities due to the combination of nonlinear terms,

and thus, multiple local minima arise in the model. Moreover, to guarantee we are within limits, we are required to rigorously make sure we are including all possible solutions. In this sense, we want our computations to rigorously determine the optimal solution by obtaining mathematically verified upper and lower bounds on the global optima.

This can be achieved if we obtain bounds on the dynamic variables that are mathematically verified to be within a safe operating range. In this context, the term "mathematically verified" means that we are able to include all the solutions within upper and lower bounds. For example, interval amounts can represent ranges of values from a lower to an upper endpoint and they include all the values within a range because the lower and the upper endpoint are rounded downwards and upwards, respectively.

Sequential and simultaneous approaches have been used for the solution of the deterministic global optimisation problem for dynamic systems. In the first, the dynamic system is integrated and in turn the objective function and gradients are evaluated. In the second, the dynamic system is converted to a set of algebraic equations by using collocation-based discretisations leaving a fully algebraic nonlinear programming (NLP) problem. Stochastic methods have also been used, although they are not considered in this chapter because they are not able to provide a guarantee that the global optima have been found.

Several chemical engineering researchers have devoted their efforts to solve the guaranteed global optimisation problem for dynamic systems. They have used complete search methods such as branch and bound frameworks to make sure no solution is left out and to focus on finding bounds for the dynamic variables. Therefore, their algorithms rigorously find all global optima within bounds or are at least able to provide a theoretical guarantee that the global optima have been found. A branch and bound framework was used with the αBB method [1] in a sequential approach and applied to four different optimal control problems including the optimal control of batch and semi-batch processes [19], and to parameter estimation problems to determine reaction kinetic constants from time series data [18]. In principle, the method of [18, 19] provides a guaranteed global optimum. The rig-

orous underestimators needed are hard to obtain and here, only sampling approaches were proposed. Another sequential approach of implementing a branch and bound framework was developed with a convex underestimating procedure [52] which they applied to parameter estimation, chemical kinetics and modelling and optimal control problems. Some years later, again using a sequential approach and a branch and bound framework, Papamichail and Adjiman [53] used McCormick relaxations and "constant and affine" bounds in the solution of parameter estimation and optimal control problems. The approach used by Papamichail and Adjiman [52, 53] is computationally expensive in constructing tight affine underestimators and overestimators. Singer and Barton [73] presented a sequential approach using another branch and bound framework for problems with an integral objective function. This algorithm implements differential inequalities and McCormick relaxations to construct the convex/concave relaxations and the method was applied to parameter estimation and optimal control problems. Rauh et al. [62] presented a global optimisation method for discrete-time and continuous-time systems applied to a mechanical positioning system. In the continuous-time system, their algorithm uses a prototype implementation of an interval extension of Taylor series as a verified solver. The guaranteed solution has also been considered for mixed-integer dynamic optimisation. Chachuat et al. [14] presents a review on these methods focusing on systems with embedded ODEs and branch and bound frameworks. They stress out the importance of tight state relaxations and the use of heuristics in the global optimisation algorithm. The dynamic optimisation problem has also been addressed using Taylor Models in a sequential approach and a branch and bound framework with focus on the tightness of the ODEs state bounds. This method uses Taylor Models method with an interval remainder term and was applied to several parameter estimation problems [32]. Later, they applied the same method but this time with a branch and reduce approach using a domain reduction technique Lin and Stadtherr [33] and the applications were an optimal control and a final time formulation of the oil shale pyrolysis problems. The later method was extended to account for inequality path constraints in a rigorous way [83] and was applied to three semi-batch reactor problems.

Only a few research groups have worked on this problem and they have mainly made use of

Table 6.1: Number of state variables and system parameters in global optimisation for dynamic systems

Reference	**Maximum number of state variables**	**Maximum number of decision variables**
Esposito and Floudas [19]	7	1
Esposito and Floudas [18]	3	5
Papamichail and Adjiman [52]	2	3
Papamichail and Adjiman [53]	2	3
Singer and Barton [73]	4	3
Lin and Stadtherr [32]	2	4
Lin and Stadtherr [33]	5	5
Zhao and Stadtherr [83]	4	8

a branch and bound framework in a sequential approach. The available methods for solving dynamic systems to global optimality are only able to address low dimensional problems. According to Table 6.1, of the order of 2-7 state variables and 1 to 8 decision variables.

In these methods, one of the main challenges to overcome is the construction of tight and efficient bounds for the dynamic constraints. This problem arises because there is overestimation present in the construction of the bounds which causes the bounds to be overconservative (yielding to poor objective function and gradient evaluation) or in the worst case they tend to $\pm\infty$ and the algorithm has to be stopped.

Bounds on the dynamic variables accumulate overestimation because usually some form of overapproximation (such as intervals or relaxations) is used to guarantee that all the solutions are included. However, the set or relaxation used is merely a representation of the true solution set and when operations with overestimated values are propagated (in an integration algorithm for example), the result can be extremely overconservative.

In this work, we propose to use interval contracting methods in a verified integration method in order to obtain tight and efficient bounds for global optimisation in dynamic systems applications.

6.2 Problem formulation

We assume that the system can be described by an ODE model $\dot{\mathbf{y}}(t) = \mathbf{f}(t, \mathbf{y}(t), \boldsymbol{\theta})$, with $\mathbf{y}$, the vector of state variables and $\boldsymbol{\theta}$, the vector of system parameters. In this chapter, the bold type notation is adopted to indicate vector-valued quantities and square brackets are used for interval-valued quantities unless otherwise specified. The lower and upper endpoints of an interval $[x]$ are specified by $[\underline{x}, \overline{x}]$, the width or diameter of an interval is given by $w([x]) = \overline{x} - \underline{x}$ and the midpoint by $\hat{x} = (\overline{x} - \underline{x})/2$. A dynamic optimisation problem involving a dynamic model can be formulated as

$$\begin{aligned} \min_{\boldsymbol{\theta}} \ & \phi(\mathbf{y}(t_i, \boldsymbol{\theta}),\ \boldsymbol{\theta};\ i = 1, \ldots, ns) \\ s.t. \ \ & \dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y}(t), \boldsymbol{\theta}) \\ & \mathbf{y}(t_0, \boldsymbol{\theta}) = \mathbf{y}_0(\boldsymbol{\theta}) \\ & t \in [t_0, t_f] \\ & \boldsymbol{\theta} \in [\boldsymbol{\theta}] \end{aligned} \tag{6.1}$$

where ϕ is the objective function, $[\boldsymbol{\theta}] = [\overline{\boldsymbol{\theta}}, \underline{\boldsymbol{\theta}}]$ are the decision variables, an interval vector where $\overline{\boldsymbol{\theta}}$ and $\underline{\boldsymbol{\theta}}$ are lower and upper endpoints, respectively.

The dynamic simulation problem that is being solved within the optimisation problem is as follows:

$$\dot{\mathbf{y}}(t) = \mathbf{f}(t, \mathbf{y}(t), \boldsymbol{\theta}),\ \mathbf{y}(t_0, \boldsymbol{\theta}) = \mathbf{y}_0(\boldsymbol{\theta}),\ \mathbf{y}_0(\boldsymbol{\theta}) \in [\mathbf{y}_0],\ \boldsymbol{\theta} \in [\boldsymbol{\theta}] \tag{6.2}$$

where $t \in [t_0, t_f]$, $\boldsymbol{\theta}$ represent the time-invariant parameters with $[\boldsymbol{\theta}] = [\underline{\boldsymbol{\theta}}, \overline{\boldsymbol{\theta}}]$, $\mathbf{y}$ represent the vector of state variables, $\mathbf{y}_0$ are the initial conditions at time t_0 with $[\mathbf{y_0}] = [\underline{\mathbf{y}_0}, \overline{\mathbf{y}_0}]$. Here, $\mathbf{f}$ is assumed to be $(k-1)$ times continuously differentiable with respect to the state variables. The parameters of the simulation problem correspond to the decision variables of the optimisation problem.

This work considers a sequential approach in which the dynamic constraints need to be integrated in order to evaluate the objective function and gradients. When a sequential

approach is used, a verified ODE method integrates the dynamic part of the optimisation problem leaving a problem only constrained by the system parameters $\boldsymbol{\theta}$. In the global optimisation algorithm (Table 6.3), two calls to the verified integration method are done per iteration. The tightness of the bounds from the verified integration method directly affects the global optimisation as we evaluate the objective function and gradients with these. Hence, it is desirable to obtain the tightest bounds possible.

6.3 Enclosing the Solutions of ordinary differential equations

The verified simulation algorithm used in the global optimisation algorithm is the same as the one used in Chapter 3. The global optimisation algorithm is presented in Table 6.2. In the algorithm, an interval Taylor series method is used in collaboration with the fixed-point Newton/Gauss-Seidel (Section 3.4.2) contractor which resulted to have better performance then the Krawczyk contractor (Section 3.4.2) in Chapter 3. The algorithm proceeds by carrying out an integration using an interval Taylor series algorithm (Section 3.3). In this integration, an implicit function is formulated which is seen as a constraint propagation problem in which the interval contractors can be applied.

The algorithm is able to compute verified bounds on the solutions of ODE systems where the initial conditions and/or system parameters can be specified to be uncertain by using a bounded range. These uncertain values represent the decision variables in the optimisation problem. Upon termination, the algorithm yields verified bounds that include the solution of the dynamic system. It also yields the enclosure of the trajectories first order sensitivity equations (the gradient of the ODEs with respect to the parameters). Finally the information obtained serves as input to the global optimisation algorithm (Table 6.3) which in turn computes bounds on the objective function and the gradient of the objective function.

Table 6.2: Interval Taylor series with fixed-point contractors algorithm

Initialise: $\mathbf{A}_0 = \mathbf{I}$, $[\mathbf{\Gamma}_0] = [\mathbf{y}_0] - \hat{\mathbf{y}}_0$

`ITSContractor`(`In`: $\mathbf{A}_j$, $[\mathbf{\Gamma}_j]$, h_j, $[\tilde{\mathbf{y}}_j]$, $[\mathbf{y}_j]$, $[\boldsymbol{\theta}]$; `Out`: $\mathbf{A}_{j+1}$, $[\mathbf{\Gamma}_{j+1}]$, $[\mathbf{y}_{j+1}]$)

begin

$\mathbf{u}_{j+1} = \hat{\mathbf{y}}_j + \sum_{i=1}^{k-1} h_j^i \mathbf{f}^{[i]}(\hat{\mathbf{y}}_j, \hat{\boldsymbol{\theta}})$

$[\mathbf{z}_{j+1}] = h_j^k \mathbf{f}^{[k]}([\tilde{\mathbf{y}}_j], [\boldsymbol{\theta}])$

$[\mathbf{S}_{j+1}^y] = \mathbf{I} + \sum_{i=1}^{k-1} h_j^i \frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{y}}([\mathbf{y}_j], [\boldsymbol{\theta}])$

$[\mathbf{S}_{j+1}^\theta] = \sum_{i=0}^{k-1} h_j^i \frac{\partial \mathbf{f}^{[i]}}{\partial \boldsymbol{\theta}}([\mathbf{y}_j], [\boldsymbol{\theta}])$

$\mathbf{Q}_j \mathbf{R}_j = \hat{\mathbf{S}}_{j+1}^y \mathbf{A}_j$

$\mathbf{A}_{j+1} = \mathbf{Q}_j$

$[\mathbf{\Gamma}_{j+1}] = \mathbf{A}_{j+1}^{-1}([\mathbf{z}_{j+1}] - \hat{\mathbf{z}}_{j+1}) + \mathbf{A}_{j+1}^{-1}([\mathbf{S}_{j+1}^y]\mathbf{A}_j)[\mathbf{\Gamma}_j] + (\mathbf{A}_{j+1}^{-1}[\mathbf{S}_{j+1}^\theta])([\boldsymbol{\theta}] - \hat{\boldsymbol{\theta}})$

$[\mathbf{y}_{j+1}] = \mathbf{u}_{j+1} + ([\mathbf{S}_{j+1}^y]\mathbf{A}_j)[\mathbf{\Gamma}_j] + [\mathbf{S}_{j+1}^\theta]([\boldsymbol{\theta}] - \hat{\boldsymbol{\theta}}) + [\mathbf{z}_{j+1}]$

Newton/Gauss-Seidel contractor:

$\mathbf{g}(\hat{\mathbf{y}}_j, [\boldsymbol{\theta}]) = \mathbf{A}_{j+1}[\mathbf{\Gamma}_{j+1}](\hat{\mathbf{y}}_j, [\boldsymbol{\theta}])$

$[\mathbf{y}_{j+1}] \supseteq [\mathbf{y}_{j+1}]_N =$`NGSContractor`$([\mathbf{y}_{j+1}], [\mathbf{S}_{j+1}^y], \mathbf{g}(\hat{\mathbf{y}}_j, [\boldsymbol{\theta}]))$

or

$[\mathbf{y}_{j+1}] \supseteq [\mathbf{y}_{j+1}]_N =$`KContractor`$([\mathbf{y}_{j+1}], [\mathbf{S}_{j+1}^y], \mathbf{g}(\hat{\mathbf{y}}_j, [\boldsymbol{\theta}]))$

$[\mathbf{y}_{j+1}] \leftarrow [\mathbf{y}_{j+1}] \cap [\mathbf{y}_{j+1}]_N$

end

6.4 Global Optimisation using Interval Taylor Series with Overestimation Reduction

The main objective of the present section is to describe the algorithm to find the global optima using the verified integration method with contractors in a sequential approach. As far as the authors know, this is the first time an interval Taylor series method with a Newton/Gauss-Seidel contractor is used in a sequential approach to solve dynamic optimisation problems to global optimality. When a sequential approach is used a verified ODE method is applied to the dynamic part of the optimisation problem leaving a problem only constrained by the system parameters $\boldsymbol{\theta}$. The spatial search procedure used was similar to a standard branch and bound algorithm by Moore et al. [44]. The global optimisation method considers a problem of the form:

$$\begin{aligned} &\min_{\boldsymbol{\theta}} \phi(\boldsymbol{\theta}) \\ &s.t.\, \boldsymbol{\theta} \in [\boldsymbol{\theta}] \end{aligned} \tag{6.3}$$

The global optimisation method that has been used in the numerical experiments is given in Table 6.3. In this algorithm, each time a box is branched, new bounds are needed and so the bounding routine (ITS method with contractors) is called for each box. This call represents the most expensive part in the branch and bound framework. Therefore it is needed to reduce the number of calls by discarding as many boxes as possible prior to the bounding step. A condition that tests whether or not zero is contained in the gradient of the objective function is being used in this algorithm in which we need to compute the first order sensitivity equations.

$$\frac{\partial}{\partial t}\left(\frac{\partial \mathbf{y}}{\partial \boldsymbol{\theta}}\right) = \frac{\partial \mathbf{f}}{\partial \mathbf{y}}\frac{\partial \mathbf{y}}{\partial \boldsymbol{\theta}} + \frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}} \tag{6.4}$$

The first order sensitivity equations are integrated in a verified way together with the ODE system.

Table 6.3: Global optimisation algorithm

`GOalgorithm`(`In`: ϕ^*_{local}, $[\boldsymbol{\theta}]_0$, ε; `Out`: $[\phi*]$, C, L)

Initialise: $[\boldsymbol{\theta}] = [\boldsymbol{\theta}]_0$, $\phi_{ub} = \phi^*_{local}$ or $\phi_{ub} = \phi(\hat{\boldsymbol{\theta}})$, $L = \varnothing$, $C = \varnothing$
while $L \neq \varnothing$
  Bisect $[\boldsymbol{\theta}]$ such that $w([\boldsymbol{\theta}]_i) = w([\boldsymbol{\theta}]) = max_{1\leq i\leq n} w([\boldsymbol{\theta}]_i) : [\boldsymbol{\theta}] = [\boldsymbol{\theta}]^{(1)} \cup [\boldsymbol{\theta}]^{(2)}$
  Call `ITSContractor`($[\mathbf{y}_0]$,$[\boldsymbol{\theta}]^{(1)}$) to evaluate $\phi([\boldsymbol{\theta}]^{(1)})$ and $\frac{\partial\phi}{\partial\theta}([\boldsymbol{\theta}]^{(1)})$
  Call `ITSContractor`($[\mathbf{y}_0]$,$[\boldsymbol{\theta}]^{(2)}$) to evaluate $\phi([\boldsymbol{\theta}]^{(2)})$ and $\frac{\partial\phi}{\partial\theta}([\boldsymbol{\theta}]^{(2)})$
  $\phi_{ub} = \min\{\phi_{ub}, \overline{\phi}(\hat{\boldsymbol{\theta}}^{(1)}), \overline{\phi}(\hat{\boldsymbol{\theta}}^{(2)})\}$
  if $\max\{\overline{\phi}([\boldsymbol{\theta}]^{(1)}), \overline{\phi}([\boldsymbol{\theta}]^{(2)})\} - \min\{\underline{\phi}([\boldsymbol{\theta}]^{(1)}), \underline{\phi}([\boldsymbol{\theta}]^{(2)})\} < \varepsilon$ **then**
    if $0 \notin \frac{\partial\phi}{\partial\theta}([\boldsymbol{\theta}]^{(1)})$ **then** Discard $[\boldsymbol{\theta}]^{(1)}$
    else if $0 \notin \frac{\partial\phi}{\partial\theta}([\boldsymbol{\theta}]^{(2)})$ **then** Discard $[\boldsymbol{\theta}]^{(2)}$
    end if
    Place $[\boldsymbol{\theta}]^{(1)}$ and $[\boldsymbol{\theta}]^{(2)}$ into C in order
    if $L \neq \varnothing$ **then**
      Remove the first item from L and place its box into $[\boldsymbol{\theta}]$
      if $\underline{\phi}([\boldsymbol{\theta}]) > \phi_{ub}$ **then**
        return with $\underline{\phi}$ equal to lower bound on the first box in C
        and with the lists C and L
      end if
    else
      return with $\underline{\phi}$ equal to lower bound on the first box in C
      and with list C
    end if
  else
    if $0 \notin \frac{\partial\phi}{\partial\theta}([\boldsymbol{\theta}]^{(1)})$ **then** Discard $[\boldsymbol{\theta}]^{(1)}$
    else if $0 \notin \frac{\partial\phi}{\partial\theta}([\boldsymbol{\theta}]^{(2)})$ **then** Discard $[\boldsymbol{\theta}]^{(2)}$
    end if
    Enter the items $([\boldsymbol{\theta}]^{(1)}, \underline{\phi}([\boldsymbol{\theta}]^{(1)})$ and $([\boldsymbol{\theta}]^{(2)}, \underline{\phi}([\boldsymbol{\theta}]^{(2)})$ in proper order in list L
    Set $[\boldsymbol{\theta}]$ as the argument of the first item in L (with lowest $\underline{\phi}([\boldsymbol{\theta}])$) and remove the item $([\boldsymbol{\theta}], \underline{\phi}([\boldsymbol{\theta}])$ from the list
  end if
end while

6.4.1 Implementation and Third Party Libraries

Just as in Chapter 3, the methods explained before were implemented in C++ and third party libraries were used for defining the interval type and for performing automatic differentiation. In particular the library PROFIL/BIAS was used for defining the interval type and FADBAD++ for the automatic differentiation.

Figure 6.1 represents the program used to implement a Global optimisation algorithm with ITS method with interval contractors NGS and K. In the top of the figure (file `funcivp.cpp`), the mathematical model of the initial value problem is defined, the Taylor coefficients ($\mathbf{f}^{[i]}([\mathbf{y}_j],[\boldsymbol{\theta}])$) and the Jacobian $[\mathbf{S}^y_{j+1}]$ are computed next in `tcfunc.cpp` and `jacfunc.cpp`. The information obtained in these programs is enough to compute the first and second stages which are the high order enclosure (`HOE.cpp`) and the tighter enclosure (`ITS.cpp`). The information from `jacfunc.cpp` is also input to the contractor routines Newton/Gauss-Seidel and Krawczyk, `Ncontractor,cpp` and `Kcontractor.cpp`, respectively. `profiles.cpp` is in charge of iterating the program and arranging the information of the trajectories; for example, it is able to subdivide and combine the initial intervals when necessary. When an optimisation problem needs to be addressed, the program `fosivp.cpp` computes the first order sensitivities which are useful to calculate the gradients later on. Finally, when all the trajectory and gradient information is ready, the optimisation problem is defined in `optimise.cpp`. This program is in charge of calling the integration program and performing the branching for sending the part of the decision space that needs to be checked for the minima. The program finishes with a list of interval vectors (boxes) including the global optima.

6.5 Numerical Case Studies

The following examples were solved in a sequential approach and using the interval Taylor series with the Newton/Gauss-Seidel contractor to bound the dynamic variables. The CPU times are for the same computer as in Section 3.5. The programs were also written in C++ and the third party libraries FADBAD++ [4] and Profil/Bias [29] were used for the auto-

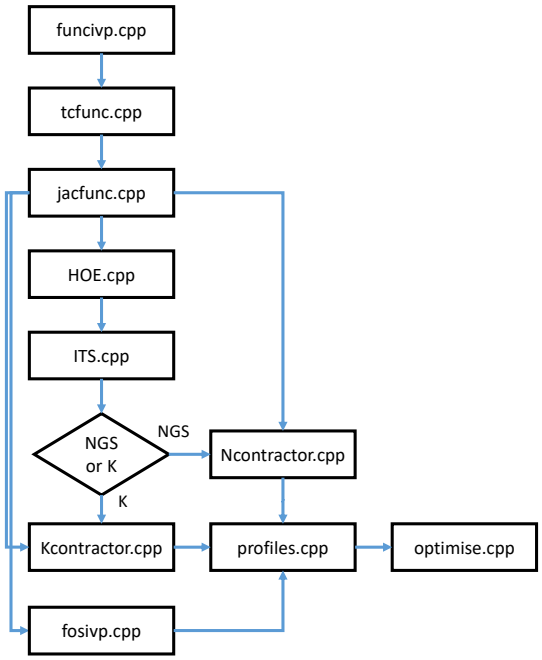


Figure 6.1: Diagram of the C++ programs to implement a Global optimisation method with the ITS method with contractors

matic differentiation and the interval arithmetic operations, respectively.

The main comparison criteria in this section are the CPU times and the number of iterations to reach the global optimum. Two other works have been used to compare the present work. In order to provide a fair comparison, the times reported by Lin and Stadtherr [32] and Lin and Stadtherr [33] have been adjusted by multiplying the time it takes VSPODE to perform a simulation of the ODE system by the number of iterations reported by each of the works. We believe this provides a better comparison since the simulations are carried out in the same machine. In order to adjust the times of Singer and Barton [73] and Papamichail and Adjiman [53], we use the relation used by Lin and Stadtherr [31] in which according to the SPEC benchmark, the times of the computers used by Lin and Stadtherr [31], Lin and Stadtherr [32] and Lin and Stadtherr [33] correspond to roughly half of the time reported in Singer and Barton [73] and to roughly 0.128 times the time reported in Papamichail and Adjiman [53].

6.5.1 First order irreversible series reaction

A first-order irreversible chain reaction taken from Tjoa and Biegler [79] considers the following reaction

$$A \xrightarrow{k_1} B \xrightarrow{k_2} C$$

The algorithm presented in Table 6.3 is applied to the parameter estimation problem with two parameters and two dynamic variables. The experimental data has been taken from Esposito and Floudas [18]. The problem has been solved to global optimality using an absolute tolerance $\varepsilon_{abs} = 10^{-4}$ in 0.40 seconds. Lin and Stadtherr [32] solved the problem with a relative tolerance $\varepsilon_{rel} = 10^{-3}$ and exactly ($\varepsilon = 0$) in 0.023 and 0.059 seconds, respectively. The machine used was an Intel Pentium 4 3.2 GHz. Singer and Barton [73] solved the problem with an absolute tolerance $\varepsilon_{abs} = 10^{-4}$ in 0.036 seconds in an AMD Athlon XP2000+ 1667 MHz. However, the differential inequalities approach is used in this work in which the auxiliary system is solved by a conventional solver and hence the solution is not computationally verified.

Table 6.4 shows these results and a column with the adjusted CPU time. In the case of the work by Singer and Barton [73], the table shows two numbers in the columns CPU (s) and Iterations because they correspond to the cases without and with heuristics. Papamichail and Adjiman [53] solved the problem in 9 and 11 seconds using constant, and constant and affine underestimation schemes, respectively. The tolerance they used was $\varepsilon_{rel} = 10^{-7}$ in an UltraSPARC-II 2×360 MHz.

$$\begin{aligned}
\min_{\mathbf{k}} \phi &= \sum_{j=1}^{10} \sum_{i=1}^{2} (x_i(t_j) - x_i^{exp}(t_j)) \\
s.t.\ \ \dot{x}_1 &= -k_1 x_1 \\
\dot{x}_2 &= k_1 x_1 - k_2 x_2 \\
x_1(0) &= 1,\ x_2(0) = 0 \\
t &\in [0, 1] \\
\mathbf{k} &\in [0, 10] \times [0, 10]
\end{aligned} \tag{6.5}$$

In the first order irreversible series reaction case study, the adjusted CPU time provided by the algorithm was much smaller than the Papamichail and Adjiman [53] method; it was higher compared to the other two works and the number of iterations was the highest of all.

6.5.2 Singular control problem

The procedure from Section 6.4 is applied to a nonlinear singular control problem taken from [37]

$$\begin{aligned} \min_u \phi &= \int_{t_0}^{t_f} [x_1^2 + x_2^2 + 0.0005(x_2 + 16t - 8 - 0.1x_3u^2)^2]dt \\ s.t.\ \ \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -x_3u + 16t - 8 \\ \dot{x}_3 &= u \\ x_1(0) &= 0, \ \ x_2(0) = -1, \ \ x_3(0) = -\sqrt{5} \\ t &\in [0, 1] \\ u &\in [-4, 10] \end{aligned} \tag{6.6}$$

Table 6.5 shows the results of the singular control problem with a column containing the adjusted CPU time. As in the previous case study, in the work by Singer and Barton [73], the table shows two numbers in the columns CPU (s) and Iterations because they correspond to the cases without and with heuristics. The sequential approach was used to provide a solution for this problem.

The computational time taken for solving the problem was 1.42 seconds with an absolute tolerance of $\varepsilon_{abs} = 10^{-3}$. The problem was also solved by Lin and Stadtherr [33] in 0.02 seconds with the same absolute tolerance. It is worth mentioning that their global optimisation algorithm implements a branch and reduce approach and is able to reduce the search space. The machine used was an Intel Pentium 4 3.2 GHz. Also, Singer and Barton [73] provided a (non-verified) solution for the problem with the same tolerance in 2 and 1.8 seconds without and with heuristics. They used an AMD Athlon XP2000+ 1667 MHz to solve the problem. In the singular control problem, the adjusted CPU time of the present

algorithm resulted to be competitive as it was similar to the quadrature variable formulation of Singer and Barton [73] but is was bigger than Lin and Stadtherr [33] and the original formulation in Singer and Barton [73].

6.5.3 Oil shale pyrolysis

The optimal temperature profile in a plug flow reactor is considered. The reactions involved and the model of the problem are shown in (A.6). In the model, only components A_1 and A_2 are included and the objective is to maximise the production of A_2. Here, u is the adjustable parameter and is taken as a piecewise constant profile.

$$
\begin{array}{ll}
 & \min_{u} \ \phi = -x_2(t_f) \\
A_1 \xrightarrow{k_1} A_2 & s.t.\ \dot{x}_1 = -k_1 x_1 - (k_3 + k_4 + k_5) x_1 x_2 \\
A_2 \xrightarrow{k_2} A_3 & \dot{x}_2 = k_1 x_1 - k_2 x_2 + k_3 x_1 x_2 \\
A_1 + A_2 \xrightarrow{k_3} 2A_2 & k_i = a_i e^{\left(\frac{-b_i/R}{698.15+50u}\right)},\ i = 1, \ldots, 5 \\
A_1 + A_2 \xrightarrow{k_4} A_3 + A_2 & x_1(0) = 1, \ \ x_2(0) = 0 \\
A_1 + A_2 \xrightarrow{k_5} A_4 + A_2 & t \in [0, 10] \\
 & u \in [0, 1]
\end{array}
\tag{6.7}
$$

Table 6.6 reports the results of the oil shale pyrolysis example and it includes a column with the adjusted CPU time. Again, the columns CPU (s) and Iteration in the work by Singer and Barton [73] correspond to the cases without and with heuristics. The dynamic optimisation problem has been solved to global optimality using an absolute tolerance of $\varepsilon_{abs} = 10^{-3}$ in 5.22 seconds. The same problem was solved by [31] in 3.2 seconds using $\varepsilon_{abs} = 10^{-3}$ and an Intel Pentium 4 3.2 GHz. Singer and Barton [73] solved the problem in a non-verified manner in 27.30 and 26.20 seconds without and with heuristics, respectively. The machine used was an AMD Athlon XP2000+ 1667 MHz. In the oil shale pyrolysis problem, the adjusted CPU time of the present method was smaller than those of Singer and Barton [73] with and without heuristics and very similar to Lin and Stadtherr [33].

Table 6.4: Global optimisation results. First order irreversible series reaction ($\varepsilon_{abs} = 1 \times 10^{-4}$)

Method	**Objective function**	**Optimiser**	**CPU (s)**	**CPU (s) adjusted**	**Iterations**
This work	1.185×10^{-6}	(5.0035, 1.0000)	0.40	0.40	75
Singer and Barton [73]	1.22×10^{-6}	(5.0, 1.0)	0.036	0.017	-
Lin and Stadtherr [32]	1.185×10^{-6}	(5.0035, 1.0000)	0.023 & 0.059	0.022	4 & 2
Papamichail and Adjiman [53]	1.185×10^{-6}	(5.0035, 1.0000)	9 & 11	1.10 & 1.35	1

Table 6.5: Global optimisation results. Singular control problem ($\varepsilon_{abs} = 1 \times 10^{-3}$)

Method	**Objective function**	**Optimiser**	**CPU (s)**	**CPU (s) adjusted**	**Iterations**
This work	0.4965	(4.071)	1.42	1.42	34
Singer and Barton [73] (quadrature variable)	0.497	(4.07)	5.2 & 3.4	1.82 & 1.19	33 & 15
Singer and Barton [73] (original)	0.497	(4.07)	2.0 & 1.8	0.7 & 0.63	21 & 15
Lin and Stadtherr [33]	0.4965	(4.071)	0.02	0.014	9

Table 6.6: Global optimisation results. Oil Shale Pyrolysis ($\varepsilon_{abs} = 1 \times 10^{-3}$)

Method	**Objective function**	**Optimiser**	**CPU (s)**	**CPU (s) adjusted**	**Iterations**
This work	-0.3479	(0.231)	5.22	5.22	39
Singer and Barton [73]	-0.3480	(0.231)	27.3 & 26.2	19.67 & 18.87	115
Lin and Stadtherr [33]	-0.3479	(0.984)	3.2	4.61	21

6.6 Conclusions

The chapter has presented a global optimisation algorithm for dynamic systems in which for the first time, an interval Taylor series with Newton/Gauss-Seidel contractor is used to integrate the dynamic system and thus evaluate the objective function and constraints of the optimisation problem.

The global optimisation method performed reasonably well in two out of three case studies. In the first case study, the number of iterations turned out to be the worst of the compared works and the CPU time was only better than the method by Papamichail and Adjiman [53]. However, in the oil shale pyrolysis case study, the number of iterations was similar and the CPU time was better than the differential inequalities [73] approach and slightly lower than the Taylor models approach [33]. In the singular control problem, the CPU time resulted better than the quadrature variable formulation without heuristics of Singer and Barton [73] and slightly higher time when using heuristics. The CPU time of the present method was not better than the rest of the compared works and the number of iterations was similar to Singer and Barton [73], but not better than Lin and Stadtherr [33].

The algorithm developed in this chapter had better performance than the differential inequalities approach in two out of three case studies and very similar to the Taylor models approach. It also offers the guarantee that the global optima have been bounded in a verified way. In future work, an interesting idea would be to implement a constraint propagation contractor [27, 5] instead of a fixed-point contractor and to apply the different contracting techniques to other kind of verified integrators. The global optimisation algorithm presented in this chapter introduced a verified integration method with an interval Newton/Gauss-Seidel contractor which could potentially enhance the overestimation reduction capabilities of other methods; for example, global optimisation methods using the Taylor models or the convex/concave differential inequalities with interval contractors.

Chapter 7

Concluding Remarks and Future Research Directions

7.1 Concluding Remarks

Reducing the overestimation generated in guaranteed computations to construct enclosures of the reachable set is of high relevance in global optimisation for dynamic systems. The quality of these bounds directly affects the global optimisation algorithms since the objective function and constraints are evaluated using the state bounds.

This thesis has presented methods for enclosing the reachable set using overestimation reduction techniques in a novel way. Interval fixed-point and forward-backward contractors were applied to an interval Taylor series method and an interval Newton/Gauss-Seidel was applied to a Taylor model method.

Because of the simplicity of the interval Taylor series (ITS) method for verified integration, most of the developments in this thesis took this algorithm as a building block to explore the proposed improvements and then expand them to more sophisticated methods. As it was shown that using contractors in this algorithm, produces tighter and more efficient bounds than the method alone. However, only in one case was the method able to compete with the software package VSPODE.

The ITS method was also subject to the implementation of a forward-backward contractor to reduce the overestimation when more information about the problem in question is known. The tightness of the bounds provided by the method were better than the Taylor models approach in four cases. This method was particularly useful when dealing with affine reaction invariants and inequality path constraints. Regarding the latter, the method successfully discards part of the reachable set that is inconsistent with the constraints which could be potentially very useful in global optimisation since discarding infeasible regions early allows convergence speed up.

The interval fixed-point contractor was extended to Taylor models where the quality of the bounds was better depending on the problem. In this case, there is a trade off between the amount of overestimation obtained and the computational cost since the simulation time is approximately twice as much as the Taylor models method.

The method with fixed-point contractors and ITS was applied to global optimisation for dynamic systems. It was shown that the method is able to outperform other methods based on differential inequalities and is competitive with methods using Taylor models. The advantage of the method used is the simplicity of the algorithms used, as they lend themselves to expansion; i.e., we showed in Chapter 5 that the Taylor models method is able to collaborate with interval contractors in a similar way as in the ITS method.

In all of the methods developed in this thesis, there was the need of implementing an overestimation reduction technique at every time step. So far the fixed-point and the forward-backward constraint propagation contractors need to be iterated one or more times per time step. In order to reduce the computational time, a way to select when to apply the contractor is needed. For example, an heuristic based on the overestimation reduction achieved in the previous step or an algorithm that turns on the contractor when the condition number increases drastically.

The condition number provided an idea of how easy or how difficult a problem could be. In practice, it was observed that the higher the condition number, the smaller the time step has to be. This minimses the variability from time step to time step.
Finally, overestimation reduction techniques are needed in verified integration to address larger uncertain inputs and higher dimensional problems in global optimisation for dynamic systems.

7.2 Future Research Directions

The following research directions are proposed as encouraging future developments on these methods.

- To have the complete set of tools from this research, the algorithm for constraint propagation could be included in a Taylor models method to address optimisation problems with inequality path constraints or with other kinds of physical constraints.
- The ITS method with constrain propagation is a promising alternative when there are known constraints for the problem. The use of the approach in a global optimisation algorithm could potentially speed-up the convergence as it would quickly discard portions of the search space that do not satisfy an inequality path constraint for example.
- The use of all of the contractors would be interesting in a differential inequalities approach or in a differential inequalities with Taylor models approach.
- Unums [22] represent yet another alternative for uncertainty management. Their key advantage is their ability to use different number of bits according to the computation needs. This allows to adjust the computational expense in an automatic manner. Furthermore, unums allow an accurate representation of general intervals with open and closed endpoints, therefore removing the overestimation at the bit level. Arithmetic based on intervals is not able to represent open endpoints and thus adding some overestimation to numbers that cannot be represented exactly as decimal numbers.
- The use of model reformulation is also an encouraging direction since the dependency

problem can be tackled looking for particular model configurations in the directed acyclic graph or expression tree so as to minimise the number of variables. Appendix A presents preliminary work in this area. Expanding the library of patterns that can be transformed to an equivalent but less dependent form for model reformulation is another research direction if one wants to see the full extent of the technique.

- Due to lack of time, the implementation of the TM with Fixed-point interval contractors was not written in an efficient language. The refinement of this implementation can reduce the CPU time and it can in turn provide the improved bounding capabilities to a global optimisation algorithm. Taylor Models in global optimisation have been successful in recent works. However, we still need to investigate if it can be improved with the help of contractors.

- An optimisation algorithm that can selectively choose the number of interval contractors needs to be developed. In our practical experience the global optimisation algorithm has to be started with maximum settings (high number of contractor iterations) because the decision space is too large. One usually wants to use the minimum number of contractor iterations as they increase the computational time. As the program progresses, the decision space becomes smaller and smaller and the need for a high number of contractor iterations is also decreased. An algorithm that can select when to reduce the number of these iterations needs to be developed to reduce the computational time. This will ensure that only the first iterations are the most expensive and that the computational effort per integration is decreased as the decision space is reduced.

- Constraint propagation should be performed in directed acyclic graphs (DAGs) instead of tree expressions since the former have proven to perform better in constraint propagation.

- In order to reduce the computational time, a way to select when to apply the contractor is needed. For instance, an heuristic based on the overestimation reduction achieved in the previous step or an algorithm that turns on the contractor when the condition number increases drastically.

- An interesting research would be to find an index (*boundability index*) that can roughly describe how easy or difficult to bound a problem can be. This could be based for example on the condition number, the number of variables, the nonlinearity, etc.

Appendices

Appendix A

Model reformulation in verified simulation

Many process engineering problems have critical bounds (quality, safety or environmental) that must be satisfied at all times. In a dynamic optimisation algorithm, guaranteed bounds on the dynamic variables of the models that describe processes are needed in order to solve the problem to global optimality in a rigorous way. Hence, verified bounds are a key step in the optimisation algorithm. Recently there has been a focus on obtaining bounds which are as tight as possible while being a close representation of the reachable set. The dependency problem and the wrapping effect, which are the main contributors to the overestimation, have been tackled a number of ways. However, the construction of tight bounds is still an issue for practical applications. In this chapter, reformulation techniques for dynamic models are investigated. Emphasis is put on the reduction of the number of repetitions of a variable in the model. A search on directed acyclic graphs (DAGs) is performed using the Mathematica 10 symbolic functions and the result is used in an interval Taylor series with contractors. The effectiveness of the technique presented is demonstrated with a chemical reaction case study. The results show that the dependency problem is alleviated and in turn reduction of the overestimation is obtained.

A.1 Introduction

Verified methods of initial value problems (IVP) for ordinary differential equations (ODE) provide enclosures which are guaranteed to contain all the possible solutions of a problem (possibly subject to an uncertain value). These methods are desirable in applications such as guarnteed parameter estimation, optimal control, and safe critical applications where critical variables such as safety or environmental performance are issues of concern.

In recent years, several software packages for the verified solution of IVPs for ODEs have been developed, popular software packages include: VNODE-LP which implements the Interval Hermite-Obreschkoff with the High Order Enclosure method [45], VSPODE makes an implementation of Taylor Models and constraint propagation is used as a way to reduce overestimation [34], ValEncIA-IVP finds the validated exponential enclosure of a nonvalidated solution and uses some interval methods to reduce overestimation [63]. Nonetheless, overestimation is still an issue in these kinds of solvers because of the dependency and wrapping effect problems. Tighter and more efficient bounds are required in applications such as global optimisation with embedded ODEs or guaranteed parameter estimation. Therefore, a number of ways to deal with this problem have been developed e.g. convex and concave relaxations (McCormick relaxations) have been incorporated alongside interval bounds [66]. These relaxations have been used in the Taylor Model method in the remainder term in addition to the interval bounds [65], and for this purpose, McCormick relaxations have been applied in a method to derive an auxiliary system of ODEs that uses interval bounds [70]. Interval contractors have not been broadly applied in the overestimation reduction of validated solutions of IVPs for ODEs algorithms. This chapter gives results on the application of an interval Taylor series method with interval Newton/Gauss-Seidel contractor to several reformulations of a case study. This chapter focuses on the dependency problem. Currently, one of the best ways to deal with the dependency problem is using Taylor models since many of the operations involved are minimised due to the use of a symbolic polynomial part in this approach. However, there is still opportunity for further reduction of the overestimation if we consider model reformulation alternatives for obtaining tighter bounds.

The remainder of the chapter is organised as follows; in section A.2, the formulation of the problem to solve and preliminaries are given; in section A.3, the bounding method with overestimation is described; in section 4, the application of the reformulation technique to the case study is presented. Section 5 provides the main results on the case study and conclusions and future research directions are given in section 6.

A.2 Problem Formulation

The system can be described by an ODE model $\dot{\mathbf{y}}(t) = \mathbf{f}(t, \mathbf{y}(t), \boldsymbol{\theta})$, with $\mathbf{y}$, the vector of state variables and $\boldsymbol{\theta}$, the vector of system parameters. The mathematical form of the problem that this method aims to solve is as follows:

$$\dot{\mathbf{y}}(t) = \mathbf{f}(t, \mathbf{y}(t), \boldsymbol{\theta}),\ \mathbf{y}(t_0, \boldsymbol{\theta}) = \mathbf{y}_0(\boldsymbol{\theta}),\ \mathbf{y}_0(\boldsymbol{\theta}) \in [\mathbf{y}_0],\ \boldsymbol{\theta} \in [\boldsymbol{\theta}] \tag{A.1}$$

where $t \in [t_0, t_f]$, $\boldsymbol{\theta}$ represent the time-invariant parameters with $[\boldsymbol{\theta}] = [\underline{\boldsymbol{\theta}}, \overline{\boldsymbol{\theta}}]$, $\mathbf{y}$ represent the vector of state variables, $\mathbf{y}_0(\boldsymbol{\theta})$ are the initial conditions at time t_0 with $[\mathbf{y_0}] = [\underline{\mathbf{y}_0}, \overline{\mathbf{y}_0}]$. Here, $\mathbf{f}$ is assumed to be $(k-1)$ times continuously differentiable with respect to the state variables.

A.3 Verified Integration Method

The bounding method used in this chapter consists of two stages. The first stage is the validation of existence and uniqueness of a solution in which a suitable a priori enclosure and a time step are obtained. The second stage involves the computation of a tighter enclosure in which a high order Taylor series is used to refine the solution obtained in the first stage.

The validation of existence and uniqueness is carried out (Section 2.2.2) and an appropriate time step h_j as well as an a priori enclosure $[\tilde{\mathbf{y}}_j] \supseteq \mathbf{y}(t; t_j, [\mathbf{y}_j], [\boldsymbol{\theta}])$, for all $t \in [t_j, t_{j+1}]$ with $t_{j+1} = t_j + h_j$ are obtained by making use of the High Order Enclosure (HOE) approach

[48]. According to this approach, h_j and $[\tilde{\mathbf{y}}_j]$ must satisfy the following equation:

$$[\tilde{\mathbf{y}}_j] = [\mathbf{y}_j] + \sum_{i=1}^{k-1} [0, h_j]^i \mathbf{f}^{[i]}([\mathbf{y}_j], [\boldsymbol{\theta}]) + [0, h_j]^k \mathbf{f}^{[k]}([\tilde{\mathbf{y}}_j^0], [\boldsymbol{\theta}]) \subseteq [\tilde{\mathbf{y}}_j^0] \tag{A.2}$$

where k is the order of the Taylor series expansion, $[\mathbf{y}_j]$ is the vector of tight enclosures of the solutions with ranges in $[\tilde{\mathbf{y}}_j^0]$, $[\boldsymbol{\theta}]$ is the vector of system parameters and $\mathbf{f}^{[i]}$ are the Taylor coefficients defined according to

$$\begin{aligned} \mathbf{f}^{[0]}(\mathbf{y}, \boldsymbol{\theta}) &= \mathbf{y}_j \\ \mathbf{f}^{[i]}(\mathbf{y}, \boldsymbol{\theta}) &= \frac{1}{i} \left(\frac{\partial \mathbf{f}^{[i-1]}}{\partial \mathbf{y}} \mathbf{f} \right) (\mathbf{y}, \boldsymbol{\theta}) \text{ for } i \geq 1 \end{aligned} \tag{A.3}$$

These Taylor coefficients can be calculated using automatic differentiation (Section 2.2.1).

The second stage refines the a priori enclosure $[\tilde{\mathbf{y}}_j]$ on $t \in [t_j, t_{j+1}]$ provided in the first stage. It satisfies the next equation

$$\begin{aligned} [\mathbf{y}_{j+1}] = \overbrace{\hat{\mathbf{y}}_j + \sum_{i=1}^{k-1} h_j^i \mathbf{f}^{[i]}(\hat{\mathbf{y}}_j, \hat{\boldsymbol{\theta}})}^{\mathbf{u}_{j+1}} + \overbrace{\left\{ \mathbf{I} + \sum_{i=1}^{k-1} h_j^i \frac{\partial \mathbf{f}^{[i]}}{\partial \mathbf{y}}([\mathbf{y}_j], [\boldsymbol{\theta}]) \right\}}^{[\mathbf{S}_{j+1}^y]} ([\mathbf{y}_j] - \hat{\mathbf{y}}_j) \\ + \underbrace{\left\{ \sum_{i=0}^{k-1} h_j^i \frac{\partial \mathbf{f}^{[i]}}{\partial \boldsymbol{\theta}}([\mathbf{y}_j], [\boldsymbol{\theta}]) \right\}}_{[\mathbf{S}_{j+1}^\theta]} ([\boldsymbol{\theta}] - \hat{\boldsymbol{\theta}}) + \underbrace{h_j^k \mathbf{f}^{[k]}([\tilde{\mathbf{y}}_j], [\boldsymbol{\theta}])}_{[\mathbf{z}_{j+1}]} \end{aligned} \tag{A.4}$$

where $\mathbf{I}$ is the identity matrix and $\hat{\mathbf{y}}_j = (\underline{\mathbf{y}_j} + \overline{\mathbf{y}_j})/2$. For the sake of simplicity, we rewrite equation (A.4) as

$$[\mathbf{y}_{j+1}] = \mathbf{u}_{j+1} + [\mathbf{S}_{j+1}^y]([\mathbf{y}_j] - \hat{\mathbf{y}}_j) + [\mathbf{S}_{j+1}^\theta]([\boldsymbol{\theta}] - \hat{\boldsymbol{\theta}}) + [\mathbf{z}_{j+1}] \tag{A.5}$$

In this method, the interval matrix-vector product $[\mathbf{S}_{j+1}^y]([\mathbf{y}_j] - \hat{\mathbf{y}}_j)$ in equation (A.5) is known to be one of the main contributors of the wrapping effect [43]. Therefore, a number of methods have been developed to try to avoid direct evaluations of this matrix-vector product [47]. In this chapter, the QR factorization technique devised by [36] is used in the interval Taylor series method. This technique or a similar variation is also used in Nedialkov

[46], Lin and Stadtherr [31], Sahlodin and Chachuat [66] and Sahlodin and Chachuat [65].

The QR factorization technique consists of the substitution of the term $[\mathbf{S}^{y}_{j+1}]([\mathbf{y}_j] - \hat{\mathbf{y}}_j)$ by another term containing a matrix that induces an orthogonal coordinate system that often provides a better enclosure than the original coordinate system.

A.3.1 Reduction of the Overestimation

In this work, the Newton with Gauss-Seidel nonlinear contractor has been used as in Perez-Galvan and Bogle [54]. For more details about the contractor see Jaulin et al. [27].

A.4 Model Reformulation

The mathematical formulation of the problem studied was modified in order to minimise the repetitions of the same variables in the model. At that point, the bounding method and the contraction technique described in Section A.3 were used. The case study was simulated using the original formulation and two more reformulations in order to compare the tightness of the bounds.

These reformulations were carried out using the *Mathematica 10* functions for symbolic manipulations. Particularly, the functions: Factor, Collect and Simplify were used to factorize the expressions, collect terms with common variables and apply a library of transformations to simplify the expressions, respectively. The bounding method with overestimation reduction was used to demonstrate the effectiveness of overestimation reduction when using a model formulated differently.

A.4.1 Model Reformulation Based on Pattern Detection

The symbolic environment in *Mathematica 10* has the ability to detect a pattern in the tree of the mathematical expression. A series of patterns of expressions for which a less dependent form is known, can be detected and substituted in the tree expression. Here, by less dependent form, we mean that the reformulated expression has less repetitions in the number of the variables of interest. The next table (Table A.1) provides the first steps

Table A.1: Pattern library

Pattern	**Reformulation**
$ax + bx$	$x(a + b)$
$\frac{x}{a} + \frac{x}{b}$	$\frac{x(a+b)}{ab}$
$ax + by - cxy$	$\frac{ab}{c} - (b - cx)\left(\frac{a}{c} - y\right)$
$ax^{\alpha} + bx^{2\beta}$	$b\left[\left(x^{\alpha} + \frac{a}{2b}\right)^2 - \left(\frac{a}{2b}\right)^{2\alpha}\right]$
$\frac{x}{a + bx}$	$\frac{1}{\frac{a}{x} + b}$
$\sqrt{\frac{1+x}{1-x}}$	$\frac{1}{\tan(\frac{\arccos x}{2})}$

towards a library of useful patterns that can be substituted in the mathematical problems in order to make them less dependent. Additionally, the ranges of subexpressions involving only positive or only negative coefficients can be found exactly when the variables range over [0,1] and symmetric intervals simplify interval multiplication. Finally, we present the first steps for an algorithm that systematically reformulates the mathematical expressions to less dependent forms.

1. Mathematica symbolic functions
   (a) Simplify[f]
   (b) Collect[f,x]
   (c) Together[f]
   (d) HornerForm[f]
2. Pattern reformulation
   (a) Thread the library of patterns in the function
   (b) Substitute the patterns
3. Input scaling
   (a) Obtain an expression g by scaling the system parameters to [0,1]

(b) Obtain an expression h by rescaling the system parameters to [-0.5,0.5]

4. Intersect the width of the bounds and pick out the tighter one

A.5 Results - A Methanol to Hydrocarbons Process

A case study is presented in this section; this process model represents the conversion of methanol to several hydrocarbons. In the numerical experiments, three different formulations of the model are considered using the verified method without and with contractors. The model previously studied by Maria and Muntean [40] will be referred to as the original model.

$$\begin{aligned}
A \xrightarrow{k_1} B \qquad & \dot{z}_1 = -\left(2\theta_1 - \frac{\theta_1 z_2}{(\theta_2+\theta_5)z_1 + z_2} + \theta_3 + \theta_4\right) z_1 \\
A + B \xrightarrow{k_2} C \qquad & \dot{z}_2 = \frac{\theta_1 z_1(\theta_2 z_1 - z_2)}{(\theta_2+\theta_5)z_1 + z_2} + \theta_3 z_1 \\
C + B \xrightarrow{k_3} P \qquad & \dot{z}_3 = \frac{\theta_1 z_1(z_2 + \theta_5 z_1)}{(\theta_2+\theta_5)z_1 + z_2} + \theta_4 z_1 \\
A \xrightarrow{k_4} C \qquad & z_0 = [1,0,0] \;\; t \in [0, 1.121] \\
A \xrightarrow{k_5} P \qquad & \theta_1 = [5, 5.4] \\
A + B \xrightarrow{k_6} P &
\end{aligned} \tag{A.6}$$

where $\theta_1 = k_1$, $\theta_2 = k_2/k_3$, $\theta_3 = k_4$, $\theta_4 = k_5$ and $\theta_5 = k_6/k_3$. A substitution $b = \theta_1 z_1/((\theta_2+\theta_5)z_1 + z_2)$ as in Esposito and Floudas [18] is carried out yielding Esposito and Floudas, 2000 formulation:

$$\begin{aligned}
\dot{z}_1 &= -\theta_1 z_1 - \theta_3 z_3 - \theta_4 z_1 - \theta_2 z_1 b - \theta_5 z_1 b \\
\dot{z}_2 &= \theta_3 z_1 + \theta_2 z_1 b - z_2 b \\
\dot{z}_3 &= \theta_4 z_1 + \theta_5 z_1 b + z_2 b
\end{aligned} \tag{A.7}$$

Since the repetition of multiple state variables and parameters is still evident, the functions mentioned above from the software package *Mathematica 10* were applied. The following

reformulation is obtained and referred to as reformulation:

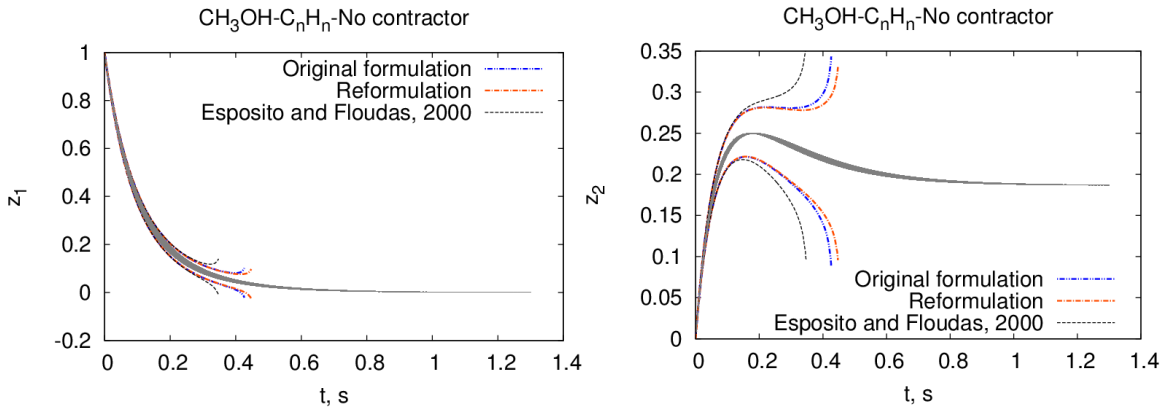
$$\begin{aligned} c &= \frac{\theta_1}{(\theta_2 + \theta_5 + z_2/z_1)} \\ \dot{z}_1 &= z_1(-\theta_1 - \theta_4 - c(\theta_2 + \theta_5)) - \theta_3 z_3 \\ \dot{z}_2 &= z_1(\theta_3 + \theta_2)c - z_2 b \\ \dot{z}_3 &= z_1(\theta_4 + \theta_5)c + z_2 b \end{aligned} \tag{A.8}$$

The verified method with overestimation reduction described in Section A.3 was used in the three models. The results are presented in Figures A.1, A.2 and A.3. In the figures, the dashed black line, the thick dashed grey line and the continuous grey line, represent the three models, respectively.

Figure A.1 represents the second state variable of the problem. Here, the method without overestimation reduction is applied to the three models. The results show that the second model (Esposito and Floudas, 2000 formulation) is the one to blow up first, however, tighter bounds are obtained at the beginning of the simulation. On the other hand, the reformulated and original model show a blow up approximately at $t = 0.4$.

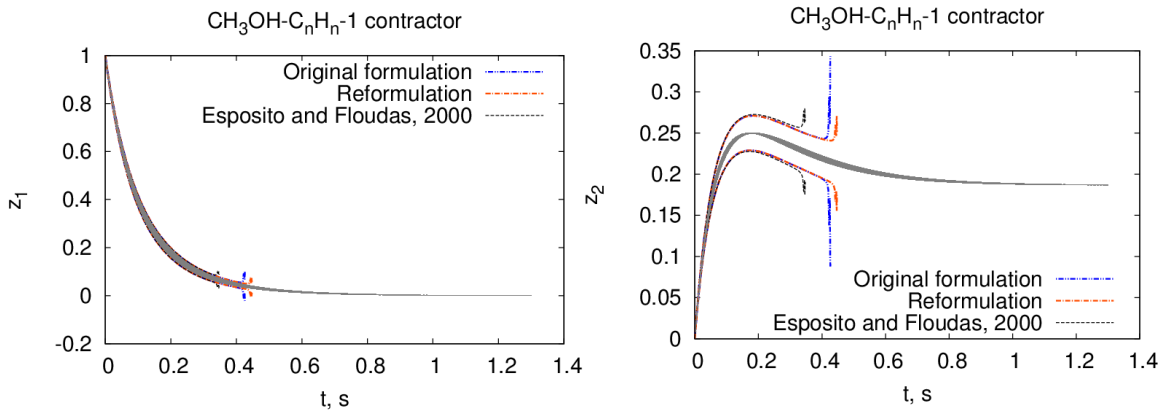
Figures A.2 and A.3 show the performance of the model reformulations when using one and two iterations of the Newton/Gauss-Seidel contractor, respectively. When using one contractor iteration, the performance of the reformulations is similar to the case without contractors. The bounds are much tighter but the same trend is observed. The three models are tight at the beginning of the simulation, but their blow up times are approximately the same as in the case with no contractor.

Finally, when two contractor iterations were used, the overestimation reduction technique managed to stabilise the bounds for the whole time horizon in the reformulated case. The original and second model performed better than in the one iteration case, but did not complete the whole time horizon.



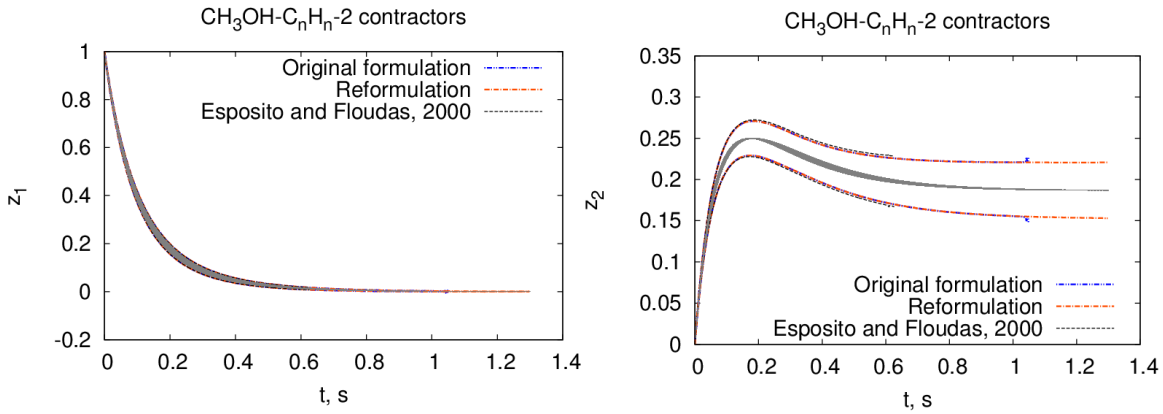
(a) (b)

Figure A.1: Methanol to hydrocarbons process without contractors



(a) (b)

Figure A.2: Methanol to hydrocarbons process with 1 Newton/Gauss-Seidel contractor iteration



(a) (b)

Figure A.3: Methanol to hydrocarbons process with 2 Newton/Gauss-Seidel contractor iterations

A.6 Conclusions

The results demonstrated the effect of reformulating a model so as to minimise the number of variable repetitions. The substitution of a common term in the model was particularly effective when using the verified method without and with the Newton/Gauss-Seidel contractor, since the bounds were stabilised in all cases. The bounds of the reformulated model (the one with less variable repetitions) turned out to be the tightest, as it always completed the longest time horizons. When two contractor iterations were used, it was the only model to complete the whole time horizon.

Using contractors for the reduction of the overestimation and a reformulated model where the number of the variables is minimal turned out to be not always the most effective in terms of bound tightness. The bounds obtained when using a common expression in the model turned out to be the best for this case study. One reason might be that this enables the recycling of a considerable amount of operations in the directed acyclic graph (DAG).

The extension of the library of common subexpressions to reduce the dependency problem is a recommended area of research to fully understand the extent of this technique. Furthermore, the use of the developed reformulation techniques presented in this thesis is also a promising direction. Model reformulation can also grow more specialised if it is used in combination with the constraint propagation approach of Chapter 4. Favourable terms that have less variable repetitions can be identified and the forward-backward contractor can deal with the simulation in the most efficient way in an automatic implementation.

The use of other kinds of reformulations based on interval numbers are another alternative. For example, a problem can be reformulated such that the use of positive intervals dominates the interval operations and in turn it could reduce the cancellation problem. Furthermore, the use of symmetric intervals is able to reduce the complexity of the multiplication rule by reducing the number of outcomes that this rule can have.

Finally, the use of unums (universal numbers) is also a promising alternative since these

are able to represent also open intervals and therefore reduce part of the overestimation.

References

[1] C. S. Adjiman, I. Androulakis, C. Maranas, and C. A. Floudas. A global optimization method, αBB, for process design. *European Symposium of Computer Aided Process Engineering*, 20(96):419–424, 1996.

[2] W. Ashworth, C. Perez-Galvan, N. Davies, and I. D. L. Bogle. Liver function as an engineering system. *American Institute of Chemical Engineers Journal*, 62(9):3285–3297, 2016. doi: 10.1002/aic.15292. URL `http://onlinelibrary.wiley.com/doi/10.1002/aic.15292/full`.

[3] S. Balendra and I. D. L. Bogle. Modular global optimisation in chemical engineering. *Journal of Global Optimization*, 45(1):169–185, feb 2009. ISSN 0925-5001. doi: 10.1007/s10898-009-9401-7. URL `http://www.springerlink.com/index/10.1007/s10898-009-9401-7`.

[4] C. Bendtsen and O. Stauning. FADBAD, a flexible C++ package for automatic differentiation. Technical report, Technical University of Denmark, Lyngby, 1996. URL `http://www2.imm.dtu.dk/{~}kajm/FADBAD/tr17{_}96.pdf`.

[5] F. Benhamou, F. Goualard, L. Granvilliers, and J. Puget. Revising hull and box consistency. *Int. Conf. on Logic Programming*, pages 230–244, 1999. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.26.463`.

[6] M. Berz and K. Makino. Verified Integration of ODEs and Flows Using Differential Algebraic Methods on High-Order Taylor Models. *Reliable Computing*, 4:361–369, 1998. URL `http://bt.pa.msu.edu/pub/papers/rdaint/rdaint.pdf`.

[7] M. Berz and K. Makino. Verified integration of ODEs and flows using differential algebraic methods on high-order Talor models, 1998.

[8] T. Bhatia and L. T. Biegler. Dynamic optimization in the design and scheduling of multiproduct batch Plants. *Industrial & Engineering Chemistry Research*, 35(7):2234–2246, 1996.

[9] A. Bompadre, A. Mitsos, and B. Chachuat. Convergence analysis of Taylor models and McCormick-Taylor models. *Journal of Global Optimization*, 57(1):75–114, 2013. ISSN 09255001. doi: 10.1007/s10898-012-9998-9.

[10] A. Cervantes and L. T. Biegler. Optimization strategies for dynamic systems. In C. A. Floudas and P. Pardalos, editors, *Encyclopedia of Optimization*, volume 1, pages 1–22. Kluwer Academic Publishers, 2nd edition, 1999.

[11] G. Chabert and L. Jaulin. Hull Consistency Under Monotonicity. In I. P. Gent, editor, *15th International Conference on Principles and Practice of Constraint Programming*, pages 188–195, Lisbon, 2009. Springer Varlag.

[12] G. Chabert, B. Neveu, and J. Ninin. Ibex library, 2007. URL `http://www.ibex-lib.org/`.

[13] B. Chachuat and M. Villanueva. Bounding the Solutions of Parametric ODEs: When Taylor Models Meet Differential Inequalities. In *22nd European Symposium on Computer Aided Process Engineering*, volume 30 of *Computer Aided Chemical Engineering*, pages 1307–1311. Elsevier, 2012. ISBN 9780444594310. doi: 10.1016/B978-0-444-59520-1.50120-2. URL `http://www.sciencedirect.com/science/article/pii/B9780444595201501202`.

[14] B. Chachuat, A. B. Singer, and P. I. Barton. Global Methods for Dynamic Optimization and Mixed-Integer Dynamic Optimization. *Industrial & Engineering Chemistry Research*, 45:8373–8392, 2006.

[15] G. F. Corliss and R. Rihm. Validating an A Priori Enclosure Using High-Order Taylor Series. (Kluwer Academic Publisher):17, 1996.

[16] P. Eijgenraam. *The solution of initial value problems using interval arithmetic: formulation and analysis of an algorithm.* Mathematisch Centrum, Amsterdam, 1981. ISBN 9061962307 9789061962304.

[17] J. A. Enszer, Y. Lin, S. Ferson, G. F. Corliss, and M. A. Stadtherr. Probability Bounds Analysis for Nonlinear Dynamic Process Models. *AIChE Journal*, 57(2):404–422, 2011. doi: 10.1002/aic.

[18] W. R. Esposito and C. A. Floudas. Global optimization for the parameter estimation of differential-algebraic systems. *Industrial & Engineering Chemistry Research*, 39(5): 1291–1310, may 2000. ISSN 0888-5885. doi: 10.1021/ie990486w. URL `http://pubs.acs.org/doi/abs/10.1021/ie990486w`.

[19] W. R. Esposito and C. A. Floudas. Deterministic Global Optimization in Nonlinear Optimal Control Problems. *Journal of Global Optimization*, 17(1-4):97–126, sep 2000. ISSN 1573-2916. doi: 10.1023/A:1026578104213. URL `http://link.springer.com/article/10.1023/A:1026578104213`.

[20] Q. Fazal and A. Neumaier. Error bounds for initial value problems by optimization. *Soft Computing*, 17(8):1345–1356, 2013. ISSN 1432-7643. doi: 10.1007/s00500-013-1007-9. URL `http://link.springer.com/10.1007/s00500-013-1007-9`.

[21] A. Flores-Tlacuahuac, L. T. Biegler, and E. Saldívar-Guerra. Dynamic optimization of HIPS open-loop unstable polymerization reactors. *Industrial & Engineering Chemistry Research*, 44(8):2659–2674, apr 2005. ISSN 0888-5885. doi: 10.1021/ie049534p. URL `http://pubs.acs.org/doi/abs/10.1021/ie049534p`.

[22] J. L. Gustafson. *The end of error: unum computing.* CRC Press, 2015.

[23] J. Hetherington, T. Sumner, R. M. Seymour, L. Li, M. Varela Rey, S. Yamaji, P. Saffrey, O. Margoninski, I. D. L. Bogle, A. Finkelstein, and A. Warner. A composite computational model of liver glucose homeostasis. I Building the composite model. *Journal of the Royal Society, Interface / the Royal Society*, 9(69):701–6, 2012. ISSN 1742-5662. doi: 10.1098/rsif.2011.0783. URL `http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3284144{&}tool=pmcentrez{&}rendertype=abstract`.

[24] B. Houska, M. E. Villanueva, and B. Chachuat. A Validated Integration Algorithm for Nonlinear ODEs using Taylor Models and Ellipsoidal Calculus. In *52nd IEEE Conference on Decision and Control*, pages 484–489, Florence, Italy, 2013.

[25] L. Jaulin, I. Braems, M. Kieffer, and E. Walter. Nonlinear state estimation using forward-backward propagation of intervals in an algorithm. In W. Kraemer and J. W. Gudenberg, editors, *SCAN-Interval 2000*, pages 191–204, Karlsrhue, Germany, 2000. Kluwer Academic Publishers.

[26] L. Jaulin, I. Braems, M. Kieffer, and E. Walter. Nonlinear state estimation using forward-backward propagation of intervals in an algorithm. In W. Krämer and J. W. von Gudenberg, editors, *Scientific Computing, Validated Numerics, Interval Methods*, chapter IV, pages 191–201. Springer, 2001.

[27] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis*. Springer, London, 2001.

[28] E. Kloppenburg and E. D. Gilles. A new concept for operating simulated moving-bed processes. *Chemical Engineering & Technology*, 22(10):813–817, oct 1999. ISSN 0930-7516. doi: 10.1002/(SICI)1521-4125(199910)22:10⟨813::AID-CEAT813⟩3.0.CO;2-G. URL `http://doi.wiley.com/10.1002/(SICI)1521-4125(199910)22:10{%}3C813::AID-CEAT813{%}3E3.0.CO;2-G`.

[29] O. Knuppel. PROFIL/BIAS-A Fast Interval Library. *Computing*, 53:277–287, 1994.

[30] F. Kruckeberg. Ordinary Differential Equations. In Hansen, editor, *Topics in Interval Analysis*, pages 91–97. Clarendon Press, Oxford, 1969.

[31] Y. Lin and M. A. Stadtherr. Deterministic Global Optimization for Dynamic Systems Using Interval Analysis. *12th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics (SCAN 2006)*, pages 38–38, sep 2006. doi: 10.1109/SCAN.2006.14. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4402428`.

[32] Y. Lin and M. A. Stadtherr. Deterministic Global Optimization for Parameter Estimation of Dynamic Systems. *Industrial & Engineering Chemistry Research*, 45

(25):8438–8448, dec 2006. ISSN 0888-5885. doi: 10.1021/ie0513907. URL http://dx.doi.org/10.1021/ie0513907.

[33] Y. Lin and M. A. Stadtherr. Deterministic Global Optimization of Nonlinear Dynamic Systems. *AIChE Journal*, 53(4):866–875, 2007. doi: 10.1002/aic.

[34] Y. Lin and M. A. Stadtherr. Validated solutions of initial value problems for parametric ODEs. *Applied Numerical Mathematics*, 57(10):1145–1162, oct 2007. ISSN 01689274. doi: 10.1016/j.apnum.2006.10.006. URL http://linkinghub.elsevier.com/retrieve/pii/S0168927406002030.

[35] Lohner. On the ubiquity of the wrapping effect in the computation of error bounds. In *Perspectives in enclosure methods*, pages 201–216. Springer, 2001.

[36] R. J. Lohner. Computation of Guaranteed Enclosures for the Solutions of Ordinary Initial and Boundary Value Problems. In J. R. Cash and I. Gladwell, editors, *Computational Ordinary Differential Equations*, pages 425–435. Clarendon Press, Oxford, 1992. URL http://www.goldsztejn.com/old-papers/Lohner-1992.pdf.

[37] R. Luus. Optimal control by dynamic programming using systematic reduction in grid size. *International Journal of Control*, 51(5):995–1013, jan 1990. doi: 10.1080/00207179008934113. URL http://www.tandfonline.com/doi/abs/10.1080/00207179008934113.

[38] K. Makino and M. Berz. Remainder Differential Algebras and Their Applications. In M. Berz, C. Bischof, G. Corliss, and A. Griewank, editors, *Computational Differentiation: Techniques, Applications, and Tools*. SIAM, 1996.

[39] K. Makino and M. Berz. Taylor models and other validated functional inclusion methods. *International Journal of Pure and Applied Mathematics*, 4(4):379–456, 2003.

[40] G. C. Maria and O. Muntean. Model reduction and kinetic parameters identification for the methanol conversion to olefins. *Chemical Engineering Science*, 42(6):1451–1460, 1987.

[41] G. P. McCormick. Computability of global solutions to factorable nonconvex programs: Part I - Convex underestimating problems. *Mathematical Programming*, 10(1):147–175, dec 1976. ISSN 00255610. doi: 10.1007/BF01580665. URL http://link.springer.com/10.1007/BF01580665.

[42] R. E. Moore. *Interval Arithmetic and Automatic Error Analysis in Digital Computing.* PhD thesis, Stanford University, 1962.

[43] R. E. Moore. *Interval analysis.* Prentice-Hall, 1967. doi: 10.112/science.158.3799.365.

[44] R. E. Moore, R. B. Kearfort, and M. J. Cloud. *Introduction to Interval Analysis*, volume 22. SIAM, Philadelphia, jan 2009. ISBN 9780898716696. doi: 10.2307/2004792.

[45] S. N. Nedialko. Implementing a Rigorous ODE Solver through Literate Programming. *Mathematical Engineering*, 3:3–19, 2011. URL http://link.springer.com/chapter/10.1007/978-3-642-15956-5{_}1.

[46] N. S. Nedialkov. Implementing a Rigorous ODE Solver Through Literate Programming. In *Modeling, Design, and Simulation of Systems with Uncertainties*, volume 3, pages 3–19. Springer, 2011. ISBN 9783642159558. doi: 10.1007/978-3-642-15956-5_1.

[47] N. S. Nedialkov, K. R. Jackson, and G. F. Corliss. Validated solutions of initial value problems for ordinary differential equations. *Applied Mathematics and Computation*, 105:21–68, 1999.

[48] N. S. Nedialkov, K. R. Jackson, and J. D. Pryce. An Effective High-Order Interval Method for Validating the Existence and Uniqueness of the Solution of an IVP for an ODE. *Reliable Computing*, 7(6):449–465, 2001. URL http://link.springer.com/article/10.1023/A:1014798618404{#}.

[49] A. Neumaier. *Interval Methods for Systems of Equations.* Cambridge University Press, Cambridge, 1990. ISBN 052133196X. URL http://books.google.co.uk/books/about/Interval{_}Methods{_}for{_}Systems{_}of{_}Equation.html?id=WzZcrWmcKJYC{&}pgis=1.

[50] A. Neumaier. Taylor Forms—Use and Limits. *Reliable Computing*, 9(1):43–79, 2003. ISSN 1385-3139. doi: 10.1023/A:1023061927787. URL http://dx.doi.org/10.1023/A:1023061927787.

[51] J. Oldenburg, W. Marquardt, D. Heinz, and D. B. Leineweber. Mixed-logic dynamic optimization applied to batch distillation process design. *AIChE Journal*, 49(11):2900–2917, nov 2003. ISSN 00011541. doi: 10.1002/aic.690491120. URL http://doi.wiley.com/10.1002/aic.690491120.

[52] I. Papamichail and C. S. Adjiman. A Rigorous Global Optimization Algorithm for Problems with Ordinary Differential Equations. *Journal of Global Optimization*, (24): 1–33, 2002.

[53] I. Papamichail and C. S. Adjiman. Global optimization of dynamic systems. *Computers & Chemical Engineering*, 28(3):403–415, mar 2004. ISSN 00981354. doi: 10.1016/S0098-1354(03)00195-9. URL http://linkinghub.elsevier.com/retrieve/pii/S0098135403001959.

[54] C. Perez-Galvan and I. D. L. Bogle. Comparison between Interval Methods to Solve Initial Value Problems in Chemical Process Design. In J. J. Klemeš, P. S. Varbanov, and P. Y. Liew, editors, *24th European Symposium on Computer Aided Process Engineering*, volume 33 of *Computer Aided Chemical Engineering*, pages 1405–1410, Budapest, 2014. Elsevier. ISBN 9780444634344. doi: 10.1016/B978-0-444-63455-9.50069-6. URL http://www.sciencedirect.com/science/article/pii/B9780444634559500696.

[55] C. Perez-Galvan and I. D. L. Bogle. Deterministic Global Dynamic Optimisation using Interval Analysis. In K. V. Gernaey, J. K. Huusom, and R. Gani, editors, *12th International Symposium on Process Systems Engineering and 25th European Symposium on Computer Aided Process Engineering*, volume 37 of *Computer Aided Chemical Engineering*, pages 791–796. Elsevier, 2015. ISBN 9780444634290. doi: 10.1016/B978-0-444-63578-5.50127-4. URL http://www.sciencedirect.com/science/article/pii/B9780444635785501274.

[56] C. Perez-Galvan and I. D. L. Bogle. Dynamic Global Optimisation Methods for De-

termining Guaranteed Solutions in Chemical Engineering. In P. M. Pardalos, A. Zhigljavsky, and J. Zilinskas, editors, *Advances in Stochastic and Deterministic Global Optimization*. Springer International Publishing, 1 edition, 2016. ISBN 978-3-319-29973-0. doi: 10.1007/978-3-319-29975-4. URL http://www.springer.com/gp/book/9783319299730.

[57] C. Perez-Galvan and I. D. L. Bogle. Reduction of the overestimation in verified simulation by model reformulation. In Z. Kravanja and M. Bogataj, editors, *26th European Symposium on Computer Aided Process Engineering*, pages 1857–1862, Slovenia, 2016. Elsevier.

[58] C. Perez-Galvan and I. D. L. Bogle. Global optimisation for dynamic systems using interval analysis. *Computers & Chemical Engineering*, 2017. doi: http://dx.doi.org/10.1016/j.compchemeng.2017.02.028. URL http://www.sciencedirect.com/science/article/pii/S0098135417300923.

[59] A. U. Raghunathan, M. Soledad Diaz, and L. T. Biegler. An MPEC formulation for dynamic optimization of distillation operations. *Computers & Chemical Engineering*, 28(10):2037–2052, sep 2004. ISSN 00981354. doi: 10.1016/j.compchemeng.2004.03.015. URL http://linkinghub.elsevier.com/retrieve/pii/S0098135404000857.

[60] A. Rauh and E. Auer. Verified Simulation of ODEs and DAEs in ValEncIA-IVP. *Reliable Computing*, (15):370–381, 2011.

[61] A. Rauh, E. P. Hofer, and E. Auer. VALENCIA-IVP: A Comparison with Other Initial Value Problem Solvers. In *12th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics (SCAN 2006)*, pages 36–36. IEEE, sep 2006. ISBN 0-7695-2821-X. doi: 10.1109/SCAN.2006.47. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4402426.

[62] A. Rauh, J. Minisini, and E. P. Hofer. Interval Techniques for Design of Optimal and Robust Control Strategies. *12th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics (SCAN 2006)*,

pages 37–37, 2006. doi: 10.1109/SCAN.2006.27. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4402427.

[63] A. Rauh, M. Brill, and C. Günther. A Novel Interval Arithmetic Approach for Solving Differential-Algebraic Equations with ValEncIA-IVP. *International Journal of Applied Mathematics and Computer Science*, 19(3): 381–397, 2009. ISSN 1641-876X. doi: 10.2478/v10006-009-0032-4. URL http://www.degruyter.com/view/j/amcs.2009.19.issue-3/v10006-009-0032-4/v10006-009-0032-4.xmlhttp://dl.acm.org/citation.cfm?id=1720066.

[64] S. M. Rump. Verification methods: Rigorous results using floating-point arithmetic. *Acta Numerica*, pages 287–449, 2010.

[65] A. M. Sahlodin and B. Chachuat. Convex/concave relaxations of parametric ODEs using Taylor models. *Computers & Chemical Engineering*, 35:844–857, 2011.

[66] A. M. Sahlodin and B. Chachuat. Discretize-then-relax approach for convex/concave relaxations of the solutions of parametric ODEs. *Applied Numerical Mathematics*, 61(7):803–820, jul 2011. ISSN 01689274. doi: 10.1016/j.apnum.2011.01.009. URL http://linkinghub.elsevier.com/retrieve/pii/S0168927411000316.

[67] H. Schichl and A. Neumaier. Interval analysis on directed acyclic graphs for global optimization. *Journal of Global Optimization*, 33(4):541–562, 2005. ISSN 09255001. doi: 10.1007/s10898-005-0937-x.

[68] J. K. Scott and P. I. Barton. Tight, efficient bounds on the solutions of chemical kinetics models. *Computers & Chemical Engineering*, 34(5):717–731, 2010. URL http://www.sciencedirect.com/science/article/pii/S0098135409003068{#}.

[69] J. K. Scott and P. I. Barton. Bounds on the reachable sets of nonlinear control systems. *Automatica*, 49(1):93–100, 2013. URL http://www.sciencedirect.com/science/article/pii/S0005109812004839.

[70] J. K. Scott and P. I. Barton. Improved relaxations for the parametric solutions of ODEs using differential inequalities. *Journal of Global Optimization*, 57(1):143–176,

may 2013. ISSN 0925-5001. doi: 10.1007/s10898-012-9909-0. URL http://link.springer.com/10.1007/s10898-012-9909-0.

[71] J. K. Scott, M. D. Stuber, and P. I. Barton. Generalized McCormick relaxations. *Journal of Global Optimization*, 51(4):569–606, feb 2011. ISSN 0925-5001. doi: 10.1007/s10898-011-9664-7. URL http://www.springerlink.com/index/10.1007/s10898-011-9664-7.

[72] J. K. Scott, B. Chachuat, and P. I. Barton. Nonlinear convex and concave relaxations for the solutions of parametric ODEs. *Optimal Control Applications and Methods*, 34: 145–163, 2013.

[73] A. B. Singer and P. I. Barton. Global Optimization with Nonlinear Ordinary Differential Equations. *Journal of Global Optimization*, 34(2):159–190, feb 2006. ISSN 0925-5001. doi: 10.1007/s10898-005-7074-4. URL http://link.springer.com/10.1007/s10898-005-7074-4.

[74] A. B. Singer and P. I. Barton. Bounding the Solutions of Parameter Dependent Nonlinear Ordinary Differential Equations. *SIAM Journal on Scientific Computing*, 27(6):2167–2182, jan 2006. ISSN 1064-8275. doi: 10.1137/040604388. URL http://epubs.siam.org/doi/abs/10.1137/040604388.

[75] A. B. Singer, J. W. Taylor, P. I. Barton, and W. H. Green. Global dynamic optimization for parameter estimation in chemical kinetics. *Journal of Physical Chemistry*, 110:971–976, 2006.

[76] B. Srinivasan, D. Bonvin, E. Visser, and S. Palanki. Dynamic optimization of batch processes I. Characterization of the nominal solution. *Computers & Chemical Engineering*, 27(1):27–44, 2003. ISSN 00981354. doi: 10.1016/S0098-1354(02)00117-5.

[77] O. Stauning and C. Bendtsen. FADBAD++ web page, 2003. URL http://www.fadbad.com/fadbad.html.

[78] T. Sumner, J. Hetherington, R. M. Seymour, L. Li, M. Varela Rey, S. Yamaji, P. Saffrey, O. Margoninski, I. D. L. Bogle, A. Finkelstein, and A. Warner. A composite computational model of liver glucose homeostasis. II Exploring system behaviour. *Journal of the*

Royal Society, Interface / the Royal Society, 9(69):701–6, 2012. ISSN 1742-5662. doi: 10.1098/rsif.2011.0783. URL http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3284144{&}tool=pmcentrez{&}rendertype=abstract.

[79] I. B. Tjoa and L. T. Biegler. Simultaneous solution and optimization strategies for parameter estimation of differential-algebraic equation systems. *Industrial & Engineering Chemistry Research*, 30(2):376–385, feb 1991. ISSN 0888-5885. doi: 10.1021/ie00050a015. URL http://dx.doi.org/10.1021/ie00050a015.

[80] M. E. Villanueva, B. Houska, and B. Chachuat. Unified framework for the propagation of continuous-time enclosures for parametric nonlinear ODEs. *Journal of Global Optimization*, 62(3):575–613, 2015. ISSN 15732916. doi: 10.1007/s10898-014-0235-6. URL http://dx.doi.org/10.1007/s10898-014-0235-6.

[81] M. E. Villanueva, J. Rajyaguru, B. Houska, and B. Chachuat. Ellipsoidal Arithmetic for Multivariate Systems. In K. V. Gernaey, J. K. Huusom, and R. Gani, editors, *12th International Symposium on Process Systems Engineering and 25th European Symposium on Computer Aided Process Engineering*, number June. Elsevier B.V., 2015.

[82] X. H. Vu, H. Schichl, and D. Sam-Haroud. Interval propagation and search on directed acyclic graphs for numerical constraint solving. *Journal of Global Optimization*, 45(4):499–531, 2009. ISSN 09255001. doi: 10.1007/s10898-008-9386-7.

[83] Y. Zhao and M. A. Stadtherr. Rigorous Global Optimization for Dynamic Systems Subject to Inequality Path Constraints. *Industrial & Engineering Chemistry Research*, 50(22):12678–12693, nov 2011. ISSN 0888-5885. doi: 10.1021/ie200996f. URL http://pubs.acs.org/doi/abs/10.1021/ie200996f.

[84] J. Žilinskas. Comparison of packages for interval arithmetic. *Informatica*, 2005. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.121.3295{&}rep=rep1{&}type=pdf.